BRÜCKMANN-SCHIEB

MICRO APPLICATION

10

AMSTRAD

LE LIVRE DU LECTEUR DE DISQUETTE AMSTRAD



UN LIVRE DATA BECKER

BRÜCKMANN-SCHIEB

MICRO APPLICATION

10

AMSTRAD

LE LIVRE DU LECTEUR DE DISQUETTE AMSTRAD



UN LIVRE DATA BECKER

Distribué par MICRO APPLICATION
13 RUE SAINTE CECILE
75009 PARIS

et également

EDITIONS RADIO
3 rue de l'Eperon
75006 PARIS

ISBN 2-86899-022-3

Traduction Française et mise en pages assurées par Pascal HAUSSMAN

Edité par Frédérique BEAUDONNET
Philippe OLIVIER

TABLE DES MATIERES

Chapitre 1:Introduction à la programmation du DDI-1

Chapitre 2: Programmation disquette avancée

Chapitre 3: Technique du lecteur et de la disquette

Chapitre 4: La ROM et la RAM de l'AMSDOS

Chapitre 5: Programmes et astuces pour le DDI-1

CHAPITRE 1: INTRODUCTION A LA PROGRAMMATION DU DD1-1

1.1 QUE PERMET DE FAIRE UN LECTEUR DE DISQUETTE?

L'Amstrad CPC 464 est une machine très appréciée qui se distingue par son bon rapport possibilités/prix. Elle est livrée avec un lecteur de cassette intégré.

Depuis peu existe également sur le marché l'Amstrad CPC 664, compatible vers le bas, qui possède un lecteur de disquette intégré. Le CPC 664 est en outre doté d'un autre clavier et son Basic a été étendu de quelques instructions.

Les deux versions (comme d'ailleurs tout autre ordinateur) ne disposent cependant que de ce qu'on appelle une mémoire fugitive: les données et programmes entrés dans l'ordinateur sont perdus dès qu'on coupe le courant. Pour pouvoir conserver ces informations, on a besoin de mémoires externes telles par exemple qu'un lecteur de cassette.

Les informations dans l'ordinateur sont alors converties en impulsions magnétiques et écrites sur cassette. Le procédé technique se présente exactement comme celui utilisé dans votre chaîne stéréo. Si vous écoutez une bande magnétique sur laquelle vous avez sauvegardé un programme, vous n'entendrez que des sons aigus assez désagréables.

Vous avez certainement découvert, à la lecture du manuel d'utilisation de la machine, que vous pouvez écrire sur la cassette avec deux vitesses différentes. Lors de la lecture, l'ordinateur adapte automatiquement sa vitesse de lecture à la vitesse utilisée lors de l'écriture. SPEED WRITE vous permet d'obtenir une vitesse d'écriture du lecteur de cassette de 1000 ou de 2000 bauds. Que signifie Baud? Baud est mis pour "bits par seconde". 1000 bauds signifie donc que l'ordinateur écrit, et lira plus tard, 1000 bits par seconde sur la cassette. 1000 bits, c'est-à-dire 1000/8, soit 125 octets par seconde. Avec 2000 bauds, ce sont 250 octets qui seront écrits ou lus par seconde. Pourquoi ne travaillerait-on donc pas toujours avec une vitesse de 2000 bauds? Tout simplement parce qu'à cette vitesse le danger d'une erreur de lecture/écriture est plus grand et que vous risquez de ne pas retrouver lors de la lecture la même chose que ce que vous aviez écrit. Vous voyez certainement ce que cela signifie.

Un lecteur de disquette présente l'avantage de permettre une seconde tentative de lecture lorsqu'une erreur de lecture s'est produite. Normalement, ce sont jusqu'à 10 tentatives de lecture d'un secteur qui seront effectuées avant que ne soit donné le message indiquant que le secteur n'est pas lisible. Un secteur est la plus petite unité pouvant être appelée sur la disquette. Sur votre lecteur de disquette, un secteur comprend 512 octets. Sur un lecteur de cassette, seul un dispositif mécanique compliqué permettrait de faire rembobiner la bande par programme, sans quoi aucune nouvelle tentative de lecture n'est possible. Mais sur l'Amstrad, même cela n'est pas possible.

Pour une utilisation professionnelle, le lecteur de cassette convient mal, et ce pour deux raisons principales:

1. Un lecteur de cassette est beaucoup trop lent. Les programmes professionnels sont toujours plus complets et donc toujours plus longs. Il n'est pas rare qu'ils atteignent une taille de 25 Koctets, ce qui signifie qu'on doit se résoudre à un long délai d'attente lors du chargement de tels programmes. Il faut en outre le plus souvent écrire et lire également des données, ce qui nous amène déjà au second inconvénient:

2. Il est très difficile de charger des fichiers bien définis à partir de la cassette. Un fichier représente soit un programme sauvegardé, soit des données sauvegardées. Si vous avez sauvegardé plusieurs programmes sur cassette, vous devez soit, ce qui est compliqué et très peu précis, faire défiler la bande et essayer ensuite de charger le fichier, soit laisser l'ordinateur chercher le fichier voulu, ce qui signifie toutefois qu'il doit alors repasser sur tous les autres fichiers figurant sur la bande avant le fichier recherché. Vous n'avez pas sur une cassette d'accès direct à chaque fichier figurant sur la cassette. Les spécialistes parlent de "Random access".

Les informations arrivent séquentiellement, c'est-à-dire octet par octet ou caractère par caractère et vous ne pouvez commander le déroulement de cette procédure. Lorsque vous voulez lire le 10ème enregistrement d'un fichier, il vous faut d'abord lire les 9 enregistrements précédents. Avec l'accès direct, vous pouvez dire directement "je voudrais lire le 10ème enregistrement".

Il est possible de remédier à ces inconvénients du travail sur cassette: les disquettes offrent en effet une plus grande vitesse de travail et

également une manipulation plus simple.

Sur les grosses installations, on utilise des disques durs qui sont les frères aînés des disquettes. Un disque dur peut contenir jusqu'à 50 Moctets (50 Mégas = 50 millions). Toutefois on copie aujourd'hui encore des données importantes sur les bandes magnétiques, plus lentes, lorsqu'on veut archiver des données. Les bandes magnétiques conviennent parfaitement dans ce cas car elles occupent moins de place et sont moins coûteuses.

Mais revenons au CPC et au travail sur disquette:

"Bienvenue dans le cercle des heureux possesseurs d'un AMSTRAD DDI-1. Vous verrez vite que votre décision était la bonne et vous ne regretterez pas votre investissement."

C'est ce que vous promet le manuel d'utilisation de votre lecteur de disquette AMSTRAD DDI-1. Nous voulons avec cet ouvrage vous convaincre de la justesse de cette phrase d'introduction en vous montrant les possibilités que recèle ce lecteur de disquette.

1.2 LA DISQUETTE

1.2.1 L'évolution

Il y encore quelques années, la plupart des disquettes avaient un format de 8 pouces. Ces disquettes se révélèrent cependant peu pratiques; un peu plus tard furent alors développées les disquettes 5 pouces 1/4 que l'on utilise aujourd'hui sur presque tous les ordinateurs familiaux ou personnels (IBM, Apple, Commodore, Sirius, etc...). Une révolution s'annonça avec Apple LISA et MacIntosh qui utilisent des disquettes 3 pouces et demi et le DDI-1 d'Amstrad travaille même avec des disquettes de 3 pouces. La miniaturisation avance donc visiblement également dans le domaine des mémoires externes.
Comme on produit encore très peu de disquettes trois pouces, elles sont jusqu'à présent assez chères. Mais une chute des prix ne devrait pas tarder, de sorte qu'elles ne coûteront vraisemblablement dans un an que la moitié de leur prix actuel.

Vous connaissez certainement les disquettes 5 pouces 1/4 qui se composent d'une enveloppe en plastique souple et d'un disque disposé à l'intérieur. Sur vos disquettes, ces disques sont cependant protégés, pour des raisons de sécurité, par un boîtier en plastique dur. Vous ne pouvez plus non plus atteindre avec vos doigts l'ouverture prévue pour la tête de lecture/écriture (voir figure 1 (1)). Cette ouverture est en effet dotée d'une fermeture métallique qui ne s'ouvre que lorsque la disquette est introduite dans le lecteur de disquette.

Il en va de même pour l'orifice d'index (2). Les disquettes ont un orifice d'index pour que le lecteur de disquette "sache" toujours quand une nouvelle rotation se produit. Les orifices d'index sont reconnus par un faisceau lumineux. Cet orifice permet donc en fait au lecteur de disquette de s'orienter. Il existe aussi des lecteurs de disquette qui n'ont pas besoin de cet orifice d'index mais ces lecteurs sont en général plus lents car il faut alors résoudre ce problème par programmation.

Ces deux ouvertures sont protégées par un clapet que vous pouvez, EN FAISANT TRES ATTENTION, écarter avec votre doigt. Prenez votre disquette dans la main, face B dirigée vers le haut, l'ouverture de la tête de lecture/écriture tournée vers l'extérieur. Introduisez ensuite votre ongle dans le rail de gauche jusqu'à ce que vous rencontriez un bouton en plastique blanc (3) que vous pouvez maintenant tirer vers vous jusqu'à la

butée. Vous voyez que l'orifice d'index ainsi que l'ouverture pour la tête de lecture/écriture sont maintenant libérés. Examinez maintenant directement la véritable disquette sur laquelle sont stockées les informations. Mais ne la touchez surtout pas avec vos doigts! Au milieu se trouve l'orifice de commande par lequel la disquette est prise et tournée. Si vous faites maintenant, avec beaucoup de précautions, pivoter la disquette avec votre autre main, vous pourrez voir également l'orifice d'index. Refermez maintenant la disquette, en relâchant le bouton blanc.

Il nous reste pour terminer à parler des deux ouvertures qui servent à la protection contre l'écriture (5). Sur le bord gauche de chaque disquette se trouve un orifice doté d'une petite flèche. Il sert à protéger une disquette contre l'écriture. Vous connaissez déjà ce principe pour votre lecteur de cassette. Lorsque l'orifice est fermé, vous pouvez écrire sur la disquette. Mais vous pouvez également repousser le bouton plastique, de façon à ce que l'orifice soit ouvert. Dans ce cas, il ne sera plus possible d'écrire sur la disquette, ce qui est très intéressant lorsque vous avez sur une disquette des programmes précieux dont vous voulez éviter à tout prix qu'ils ne soient effacés par mégarde. Sur les disquettes fournies par Amstrad et qui contiennent CP/M et LOGO, le bouton doit être poussé du haut vers le bas. Sur les disquettes d'autres constructeurs, les choses sont légèrement différentes mais tout aussi simples.

Vous pouvez utiliser les deux faces de vos disquettes. Pour les disquettes 5 pouces 1/4, cette possibilité était indiquée par l'inscription "Double sided"; un luxe qu'il fallait payer. Sur votre Amstrad, il est cependant évident que vous pouvez utiliser les deux faces de vos disquettes. Vous pouvez stocker 180 Koctets sur chaque face, soit 360 Koctets ou 360 000 caractères sur une disquette.

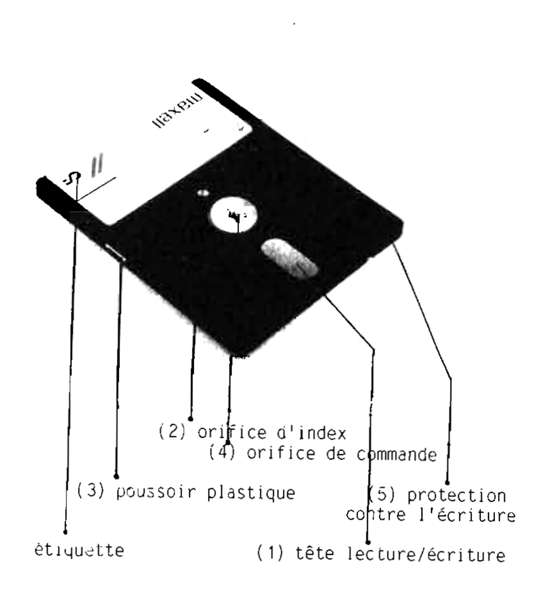
Pour placer une disquette dans le lecteur de disquette, il vous faut simplement la prendre dans la main et l'enfoncer avec précaution dans le lecteur de disquette, en suivant le sens de la flèche, jusqu'à ce que vous entendiez un claquement sourd: la disquette est alors bien en position dans le lecteur. Pour la retirer, appuyez sur le bouton qui se trouve à droite de l'ouverture prévue pour la disquette. C'est ce bouton qui permet "l'éjection" de la disquette. Mais n'actionnez jamais ce bouton lorsque votre lecteur est en train de travailler! Ceci pourrait en effet se traduire par une perte définitive de données.

Faites également attention à ce qu'aucune disquette ne se trouve dans le

lecteur de disquette aussi bien lors de la mise sous tension que lors de la mise hors tension; ceci pourrait en effet également provoquer une perte de données du fait des pointes de tension qui se produisent alors sur la tête de lecture/écriture.

Une disquette que vous venez d'acheter est encore "brute", un peu comme une feuille de papier non quadrillée. Avant qu'une disquette ne puisse ^etre utilisée, il faut la formater, un peu comme on quadrille une feuille de papier pour qu'on puisse ensuite véritablement écrire dessus.
Nous vous expliquons au chapitre 2 (CP/M) comment vous pouvez formater une disquette.

Figure 1



1.2.2 LA DISQUETTE FOURNIE AVEC LE LECTEUR

Lorsque vous achetez un DDI-1 ou bien sûr également lorsque vous achetez un CPC 664, vous recevez une disquette système. Sur cette disquette figure le système d'exploitation CP/M et sur la face B également Dr. LOGO. Le langage LOGO est maintenant très apprécié. Il avait été développé pour rendre possible à des enfants de "programmer" des ordinateurs. Si vous voulez utiliser le LOGO, sachez qu'il existe sur ce langage une littérature spécialisée assez abondante.

Dans le manuel fourni avec la disquette figurent toutes les instructions LOGO, expliquées parfois au moyen de petits exemples. Dr. LOGO signifie Digital Research Logo et il s'agit d'une version du LOGO adaptée à l'Amstrad CPC. On a augmenté le langage LOGO de quelques instructions de son de façon à ce que ces possibilités de l'Amstrad CPC puissent également être utilisées. D'autre part les touches curseur ont été également intégrées pour l'édition du programme.

1.3 QUELQUES NOTIONS SUR CP/M

1.3.1 QU'EST-CE QUE CP/M?

Sur la face A de votre disquette se trouve le système d'exploitation CP/M. CP/M signifie Control Program for Microcomputers. CP/M existe sur les ordinateurs dotés des processeurs très répandus que sont le 8080, le 8085 et le Z80. Qui possède un ordinateur travaillant sous CP/M peut accéder à une grande bibliothèque de logiciels utilisateur. CP/M présente l'avantage que très peu ou même aucunes modifications ne sont nécessaires pour faire tourner un programme existant sur un autre ordinateur. Il existe sous CP/M des tables de saut pour certaines routines (telles que la sortie sur écran, etc...) qui ont été définies une fois pour toutes. Ces tables de saut sont appelées par les programmes CP/M de sorte que les adaptations nécessaires sont ramenées au minimum. Les routines de base correspondantes sont chargées sur chaque ordinateur lors de la mise en place de CP/M.

Vous verrez que vous n'aurez pas longtemps à attendre avant que le premier logiciel CP/M, comme par exemple WORDSTAR, ne devienne également disponible sur votre CPC.

Un autre avantage vient de ce qu'un utilisateur qui a appris à utiliser CP/M peut également travailler sur n'importe quel autre ordinateur sous ce système d'exploitation. Qui a appris le Basic sur un ordinateur X est par contre encore loin de pouvoir programmer avec succès en Basic sur n'importe quel ordinateur Y. Contrairement à CP/M qui est très standardisé. Sur votre disquette figure CP/M 2.2. Vous en aurez de toute façon besoin pour formater ou copier des disquettes mais vous pouvez également l'utiliser tout autrement. Si vous voulez pour cette raison apprendre CP/M, le marché des ouvrages techniques vous offre un très large choix.

1.3.2 CHARGEMENT ET LANCEMENT DE CP/M

Une remarque préalable: lorsque vous travaillez avec votre lecteur de disquette, allumez toujours en premier le lecteur de disquette et ensuite seulement le moniteur et l'ordinateur. Sinon une procédure de test qui identifie tous les périphériques connectés enregistrera que le lecteur de disquette n'est pas connecté et toutes les instructions qui devraient

normalement être envoyées au lecteur de disquette seront adressées comme auparavant au lecteur de cassette.

Introduisez maintenant la disquette système dans le lecteur de disquette de façon à ce que son étiquette puisse être lue de l'extérieur et que la flèche avec l'inscription CP/M soit pointée vers le haut. Entrez maintenant ICPM. (Remarque: I est mis pour le trait vertical que vous obtenez à l'écran lorsque vous actionnez simultanément les touches SHIFT et .)
Maintenant CP/M est chargé de la disquette dans l'ordinateur.
Vous obtenez le message suivant:

CPM 2.2 - Amstrad Consumer Electronics plc.
A>

Dès maintenant, toutes les instructions Basic sont inopérantes. Vous pouvez en faire l'expérience en entrant par exemple:

run

CP/M vous répond:
RUN?

ce qui signifie "Mais je ne connais pas l'instruction RUN". Vous avez certainement déjà remarqué qu'un "A>" se trouve dans la première colonne. Il s'agit de ce qu'on appelle "promptsymbol" ou symbole d'interrogation. On vous indique ici que l'ordinateur attend vos ordres et qu'il est commuté sur le lecteur A. Si vous ne possédez qu'un lecteur de disquette, vous obtiendrez toujours ce message; si vous possédez deux lecteurs, vous rencontrerez également le symbole d'interrogation B>.
Mais essayez maintenant une instruction compréhensible pour CP/M:

dir

Vous obtenez instantanément sur l'écran le catalogue de la disquette. dir est mis ici pour le mot anglais signifiant catalogue, Directory.

1.3.3 FORMATAGE DES DISQUETTES

Formater signifie préparer une disquette brute pour le lecteur de disquette. Une disquette non-formatée est pour le lecteur de disquette

comme une feuille de papier sur laquelle on a écrit à l'encre blanche.

Pour formater une disquette, vous devez entrer sous CP/M l'instruction suivante:

format

Sur l'écran apparaît:

Please insert disc to be formatted into drive A then press any key

Retirez maintenant la disquette CP/M et placez dans le lecteur la disquette que vous voulez formater. Lors du formatage, toutes les informations éventuellement stockées sur une disquette sont effacées, c'est-à-dire que si vous formatez une disquette qui était déjà formatée et sur laquelle vous aviez sauvegardé des programmes ou d'autres informations, ces programmes seront perdus. C'est pourquoi il convient lors d'un formatage de se hâter avec lenteur car une fois que vous avez fait formater une disquette, il n'y a plus de contre-ordre possible. Une fois que vous avez changé la disquette, appuyez sur n'importe quelle touche et le formatage commencera aussitôt. Chaque face de la disquette est formatée avec 40 pistes qui sont disposées de façon concentrique autour de l'orifice de commande, de la piste 0 à la piste 39. La piste 0 est à l'extérieur, la piste 39 à l'intérieur. Outre les pistes, il existe cependant encore d'autres subdivisions de la disquette, les SECTEURS. La disquette est en effet subdivisée en 9 secteurs qu'on peut comparer à des parts de gâteau.
On ne peut lire la disquette que secteur par secteur. La division en pistes et secteurs se retrouve sur la plupart des systèmes de disquette.

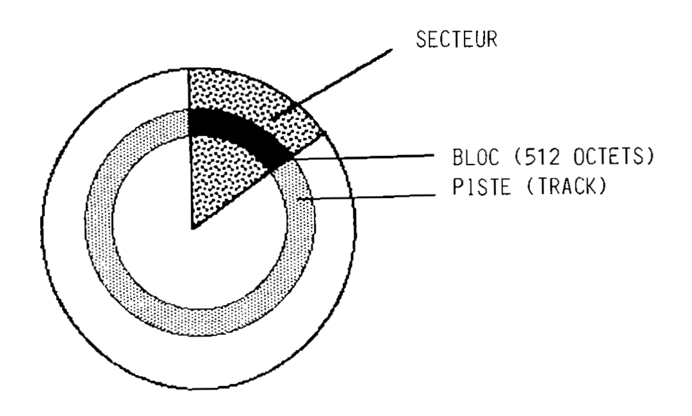


FIGURE 2

D'une façon générale, il faut distinguer les termes suivants: piste, secteur, bloc et enregistrement.

Une PISTE est, comme nous l'avons expliqué, un quarantième de la disquette. Chaque piste a 9 secteurs dont chacun comprend 512 octets.

Il existe cependant pour CP/M encore une autre subdivision, ce qu'on appelle les ENREGISTREMENTS. Un enregistrement ne comprend ni plus ni moins de 128 octets. Chaque secteur comprend donc 4 enregistrements. Cette subdivision est nécessaire pour des raisons de compatibilité avec CP/M.

AMSDOS connaît encore une autre, la dernière possibilité de subdivision, les BLOCS. Un bloc comprend 1024 octets et est donc constitué de 2 secteurs. Un bloc est la plus petite unité pouvant être appelée en Basic.

Mais revenons à notre véritable sujet, le formatage.

Après que les 40 pistes aient donc été formatées, le système vous demande:

Do you want to format another disc (Y/N):

Si vous voulez donc par exemple formater l'autre face de la disquette, répondez y pour "Yes=oui"; vous pouvez cependant également faire ainsi formater n'importe quelle autre disquette.
Le formatage peut être répété aussi souvent que vous le voulez, jusqu'à ce que vous répondiez à la question par un n pour "No=non". Le système vous demandera alors:

Please insert a CP/M system disc into drive A then press any key:

Vous pouvez maintenant actionner n'importe quelle touche, vous n'êtes pas en effet obligé de changer la disquette car le système a placé, lors du formatage, le système CP/M (mais pas les fichiers d'instructions comme par exemple l'instruction format) sur les deux pistes extérieures de la disquette.

Lorsque vous voulez formater une disquette, celle-ci ne doit pas être protégée contre l'écriture. Sinon vous obtenez le message:

Drive A:disc is write protected

Retry, Ignore or Cancel?

Ce message vous indique que la protection contre l'écriture empêche le formatage. Il existe encore d'autres messages d'erreur semblables pour lesquels vous pouvez également choisir entre:

a) Retry — Lorsque vous voulez essayer encore une fois, appuyez sur la touche "r".

b) Ignore — Si vous voulez ignorer le message d'erreur du lecteur de disquette, celui-ci poursuivra son travail. Actionnez à cet effet la touche "i". Ceci est cependant rarement conseillé car des effets inattendus peuvent alors se produire lors des instructions suivantes.

c) Cancel — En actionnant la touche "c" vous interrompez la procédure de travail en cours. C'est aussi ce que vous devriez faire dans notre exemple, de sorte à d'abord, s'il y a lieu, retirer la protection contre l'écriture.

1.3.4 COPIE SOUS CP/M

Il est très facile de copier sous CP/M le contenu d'une disquette entière sur une autre disquette. Si vous ne disposez que d'un lecteur, entrez l'instruction:

disccopy

Si vous disposez de deux lecteurs, vous pouvez utiliser l'instruction copydisc qui travaille plus vite.

Après que vous ayez entré disccopy, l'ordinateur vous demande:

Please insert source disc into drive A
then press any key

Si vous obtenez toutefois sur l'écran le message:

DISCCOPY?

cela signifie que ce n'est pas la disquette CP/M qui se trouve dans le

lecteur de disquette.

Retirez maintenant la disquette CP/M du lecteur. Si vous voulez cependant copier la disquette CP/M elle-même, ce que vous devriez absolument faire au moins une fois, laissez la disquette CP/M dans lé lecteur. Actionnez maintenant une touche quelconque.

Copying started
Reading track 0 to 7

L'ordinateur lit maintenant les 8 premières pistes (=tracks) et les charge dans l'ordinateur. Lorsque cela est terminé, vous obtenez le message:

Please insert destination disc into drive A
then press any key

Retirez maintenant la disquette source du lecteur de disquette et introduisez la disquette sur laquelle doivent être copiés les fichiers. Si votre disquette objet n'est pas encore ou si elle est incorrectement formatée, celle-ci sera d'abord formatée. D'autre part, toutes les données figurant sur la disquette seront effacées car c'est une copie intégrale de la disquette source qui est effectuée. On ne peut plus distinguer en théorie et en pratique la copie de l'original.

Après avoir placé la disquette objet dans le lecteur et après avoir actionné une touche quelconque, vous obtenez le message:

Writing track 0 to 7

Répétez la même procédure de travail pour les pistes (tracks) 8 à 15, 16 à 23, 24 à 31 et 32 à 39. La copie de la disquette est alors terminée et vous obtenez le message:

Do you want to copy another disc (Y/N):

Si vous ne voulez pas copier d'autre disquette, entrez N et suivez les instructions qui vous sont données à l'écran. Pour effectuer une autre copie, la procédure est exactement la même.

HARDCOPY D'UNE PROCEDURE DE COPIE

```
OISCCOPY

OISCCOPY V2.0

PLEASE INSERT SOURCE OISC INTO DRIVE A THEN PRESS ANY KEY..
COPYING STARTEO
READING TRACK  0
READING TRACK  1
READING TRACK  2
READING TRACK  3
READING TRACK  4
READING TRACK  5
READING TRACK  6
READING TRACK  7
PLEASE INSERT DESTINATION OISC INTO DRIVE A THEN PRESS ANY KEY..
FORMATTING WHILST COPYING
WRITING TRACK  0
WRITING TRACK  1
WRITING TRACK  2
WRITING TRACK  3
WRITING TRACK  4
WRITING TRACK  5
WRITING TRACK  6
WRITING TRACK  7
PLEASE INSERT SOURCE OISC INTO DRIVE A THEN PRESS ANY KEY
READING TRACK  8
READING TRACK  9
READING TRACK 10
READING TRACK 11
READING TRACK 12
READING TRACK 13
READING TRACK 14
READING TRACK 15
PLEASE INSERT DESTINATION OISC INTO DRIVE A THEN PRESS ANY KEY _
```

ETC...

1.3.5 COPIE AVEC DEUX LECTEURS DE DISQUETTE

Vous ne pouvez utiliser l'instruction COPYDISC que si vous avez deux lecteurs de disquette. La procédure est semblable à celle de l'instruction DISCCOPY que nous venons de décrire. L'avantage de cette instruction, comme nous l'avons déjà indiqué, est cependant que vous n'avez plus à échanger sans arrêt les disquettes source et objet.

Avec l'instruction COPYDISC également, la disquette objet est automatiquement formatée si nécessaire. Pour le reste vous avez simplement à suivre les instructions qui vous sont données à l'écran.

1.3.6 DISCCHK - VERIFICATION DES DISQUETTES

Après avoir réalisé une copie, vous pouvez également vérifier si tout a bien été copié correctement; il serait en effet rageant que votre disquette originale se détériore et que vous constatiez ensuite que la copie est également en mauvais état (ce qui peut se produire de temps en temps). Pour vérifier les disquettes source et objet, entrez l'instruction:

```
discchk
```

Suivez alors les instructions sur l'écran qui vous demandent de placer dans le lecteur la disquette originale ou la disquette de copie. Si des différences sont constatées entre la disquette originale et la disquette objet, vous obtenez le message:

```
Failed to verify destination disc correctly:
track x sector y
```

La comparaison des deux disquettes se poursuit malgré tout, de sorte que d'autres divergences éventuelles peuvent être affichées de la même façon. Ce sont toujours 8 pistes qui sont comparées consécutivement. Si vous avez deux lecteurs de disquette la comparaison s'effectuera nettement plus vite et plus simplement. Utilisez dans ce cas l'instruction:

```
chkdisc
```

Vous n'avez plus alors qu'à disposer les disquettes source et objet en vous conformant aux instructions qui vous sont données à l'écran et les

deux disquettes seront comparées exactement comme avec l'instruction discchk.
Vous pouvez par ailleurs interrompre l'exécution de toutes les instructions CP/M en tenant simultanément enfoncées les touches [CTRL] et "C". Si vous avez par exemple entré par erreur l'instruction format, vous pouvez revenir avec [CTRL]-C à la procédure d'entrée de CP/M.

Voici maintenant une liste des autres fonctions de contrôle:

[CTRL]C	=Interruption de l'instruction actuelle
[CTRL]S	=Arrête la sortie sur écran. En appuyant sur n'importe quelle touche, vous pouvez faire se poursuivre la sortie sur écran.
[CTRL]P	=Commutation sur la sortie sur imprimante, c'est-à-dire que la sortie sur écran est envoyée sur l'imprimante.
[CTRL]Z	=Fin du texte. Est utilisé par exemple lors des entrées de texte.

1.3.7 FORMATS DE FORMATAGE SPECIAUX

Lorsque vous voulez par exemple formater une disquette ne devant comporter que des données ou lorsque vous voulez utiliser une disquette exclusivement pour y stocker des programmes Basic, vous n'êtes pas obligé de copier également le système d'exploitation CP/M. CP/M est placé sur les pistes 0 et 1. Mais sur une disquette de données, vous n'avez pas besoin de CP/M, de sorte que vous gaspilleriez ainsi l'emplacement occupé par ces deux pistes. Pour éviter ce gaspillage, on peut utiliser une instruction spéciale:

format d

Lorsque vous utilisez cette instruction au lieu de format, la disquette est formatée sans recevoir le système d'exploitation CP/M. Vous avez ainsi plus de place à votre disposition pour des données ou des programmes. Voici les différentes possibilités de formatage:

format	formatage avec copie de CP/M
format d	formatage en format de données, sans CP/M
format i	formatage en format IBM
format v	formatage en format Vendor

C'est toujours la disquette se trouvant dans le lecteur de disquette standard (A) qui est formatée. Il n'est malheureusement pas possible de formater une disquette qui se trouve dans un autre lecteur.

Le format le plus usuel pour vous est le premier si vous travaillez sous CP/M ou le second si vous ne travaillez pratiquement qu'en Basic sous AMSDOS. On formate ici 9 secteurs par piste. Dans le format système, les secteurs sont numérotés de #41 à #49 (hexadécimal). Les pistes réservées sont occupées comme suit:

Piste 0, secteur #41	:secteur boot
Piste 0, secteur #42	:secteur de configuration
Piste 0, secteur #43-#47	:inutilisé
Piste 0, secteur #48,#49	:comme:
Piste 1, secteur #41-#49	:CCP et BDOS

CCP = Console Command Processor
BDOS = Basic Disk Operating System

Le format Vendor est identique au format système, sauf que le système d'exploitation CP/M n'est pas copié. Ce format est utilisé dans l'industrie des logiciels car on ne peut pas vendre CP/M avec les logiciels.

Le format DATA-ONLY

Le formatage est le même que pour le format système. 40 pistes de 9 secteurs chacune sont formatées. Les secteurs sont numérotés de #C1 à #C9. CP/M n'est pas copié, de sorte que les deux pistes supérieures, la piste 0 et la piste 1 sont disponibles pour l'utilisateur. Vous avez ainsi 2*9*512=9216 octets de plus à votre disposition.

Le format IBM

Le formatage se fait avec 8 secteurs par piste (#1-#8). Une piste est réservée. Ce format est évidemment le même que le format de disquette utilisé sur l'IBM PC sous CP/M.

Ce format ne doit être utilisé que pour une utilisation spécifique, il n'est pas sinon à recommander.

1.3.8 INSTRUCTIONS TRANSITOIRES ET RESIDENTES

Sous CP/M il faut distinguer entre ce qu'on appelle les instructions transitoires et les instructions résidentes. Les instructions résidentes sont automatiquement disponibles après le boot (c'est-à-dire le chargement général du système) de CP/M. Elles résident alors formellement dans la mémoire comme autrefois les rois dans leurs châteaux.

Il s'agit des instructions suivantes:

SAVE,A:,B:,DIR,ERA,REN,USER et TYPE.

Mais la plus grande partie, de très loin, des instructions appartiennent au groupe des instructions transitoires. Les instructions transitoires doivent être d'abord chargées à partir de la disquette, avant que l'ordinateur ne puisse les exécuter. Il s'agit des instructions:

AMSDOS, BOOTGEN, CHKDISC, CLOAD, COPYDISC, CSAVE, DISCCHK, DISCCOPY, FILECOPY, FORMAT, SETUP et SYSGEN.

Ces instructions tournent exclusivement sur l'Amstrad CPC, d'autres fournisseurs utilisent cependant des programmes utilitaires semblables sous des noms semblables ou identiques. Voici d'autres instructions CP/M transitoires standardisées:

ED, MOVCPM, PIP et STAT.

Vous trouvez ces fichiers sur votre disquette système CP/M. Les noms de fichiers sont marqués par un .COM. Vous pouvez par exemple trouver sur votre disquette le fichier FILECOPY.COM (en utilisant l'instruction DIR). Les instructions transitoires ne sont pas disponibles si elles n'ont pas été copiées avec DISCCOPY, COPYDISC, FILECOPY ou PIP.

Pour avoir une réplique exacte de votre disquette CP/M, vous devez la copier soit avec:

DISCCOPY (pour les ordinateurs avec un seul lecteur) ou avec COPYDISC (pour les possesseurs de deux lecteurs de disquette).

Vous pouvez également copier tous les fichiers transitoires avec:

FORMAT et ensuite avec FILECOPY *.*

Une autre possibilité est de ne copier que les fichiers transitoires que vous utilisez le plus souvent. C'est ainsi que vous pouvez par exemple négliger les instructions PIP, COPYDISC et CHKDISC lorsque vous n'avez qu'un seul lecteur. Il vous restera ainsi plus de place sur votre disquette.

1.3.9 COPIE DE FICHIERS AVEC FILECOPY

Pour copier différents fichiers, utilisez l'instruction:

filecopy

Si vous avez toutefois deux lecteurs de disquette, vous pouvez également utiliser l'instruction CP/M PIP.
Si vous voulez copier tous les fichiers d'une disquette, entrez:

filecopy *.*

Un nom de fichier est constitué de deux parties séparées entre elles par un point ".". La première partie est le nom de fichier proprement dit et la seconde partie est le type de fichier. La première partie peut comporter jusqu'à 8 caractères, la seconde jusqu'à 3 caractères. Cette règle ne vaut pas seulement, sur votre lecteur DD1-1, sous CP/M, mais également sous AMSDOS. Il existe par exemple les types de fichier suivants:

Type indéfini. Il pourrait s'agir par exemple d'un fichier créé en Basic qui a été ouvert sans type de fichier au moyen de l'instruction OPENOUT.

.BAS Il s'agit ici d'un programme Basic qui a été sauvegardé avec l'instruction Basic SAVE"programme" ou SAVE"programme",P ou SAVE"programme.bas",a.

.BIN Sauvegarde d'une zone de mémoire ou d'un programme machine.

.BAK Il s'agit ici de ce qu'on appelle un Backup, une copie de sécurité. Lorsque vous sauvegardez par exemple un programme sous AMSDOS et que vous utilisez un nom sous lequel un programme a déjà été sauvegardé, AMSDOS crée alors automatiquement un fichier Backup. Cela se produit pour des raisons de sécurité, pour que

vous n'effaciez par un fichier par mégarde. Mais si vous sauvegardez encore une autre fois un programme sous le même nom, l'ancien fichier Backup est effacé, le dernier fichier devient le fichier Backup et votre nouveau programme est sauvegardé sous le nom "Nom.Bas".

.COM Il s'agit ici de fichiers d'instruction dans lesquels sont sauvegardées des instructions. C'est ainsi que tous les programmes utilitaires de CP/M sont de ce type de fichier.

.DAT Il s'agit d'un fichier qui a été par exemple créé sous AMSDOS par un programme de gestion de fichier (comme dans le présent ouvrage).

.SEQ Le fichier est de type "séquentiel". Le lecteur de disquette DD1-1 dispose uniquement de ce type de fichier, tous les fichiers sont sauvegardés de façon séquentielle.

Il n'y a au fond que deux modes de sauvegarde sur lesquels sont fondés tous les autres modes. Ce sont a) la sauvegarde séquentielle et b) la sauvegarde relative de données. Le DD1-1 ne dispose de façon standard que du mode le plus simple, la sauvegarde séquentielle de données. Nous vous permettrons cependant dans le présent ouvrage de réaliser aussi des fichiers relatifs sur votre DD1-1 (voir Chapitre 5).

Par ailleurs, toute autre désignation de type de fichier est imaginable car ici n'importe quelle combinaison de trois lettres est autorisée. Les types de fichiers que nous vous avons présentés ici sont toutefois les plus usités.

1.3.10 LES WILD CARDS

Les étoiles utilisées dans l'instruction filecopy *.* sont ce qu'on appelle des wild cards (=cartes sauvages ou jokers). Ces jokers peuvent souvent vous faciliter la vie. Mais que sont donc les jokers? Il y a deux formes de jokers, les étoiles et le point d'interrogation. Si vous copiez un fichier avec l'instruction FILECOPY MATHS.BAS, le programme FILECOPY charge le fichier MATHS.BAS en mémoire et l'écrit ensuite sur la disquette objet sous exactement le même nom. Vous avez ainsi spécifié de façon très précise quel fichier devait être copié. Il y a cependant souvent des cas où vous voulez copier (ou appeler) plusieurs fichiers

différents portant des noms semblables. Supposons par exemple que vous vouliez copier exclusivement tous les programmes Basic, donc tous les programmes dont le nom se termine par un .BAS. Il vous faudrait pour cela examiner le catalogue, noter tous les fichiers dont le nom se termine par un .BAS puis copier ces fichiers les uns après les autres avec l'instruction filecopy Nom.BAS. C'est certainement assez pénible. Pour éviter cela, il y a heureusement les jokers que vous ne trouvez d'ailleurs pas uniquement sous CP/M.

Dans notre exemple, vous n'auriez plus à entrer que l'instruction suivante:

filecopy *.BAS

et tous les programmes Basic seraient copiés. L'étoile signifie "quelle que soit la première partie du nom de fichier, s'il s'agit d'un programme Basic, alors copie-le".

Seront donc ainsi copiés tous les programmes Basic.

Il est aussi possible de ne placer l'étoile qu'après un ou plusieurs caractères. Par exemple:

filecopy f*.*

signifie copie tous les fichiers qui commencent par un "f", quel que soit leur type de fichier. Les fichiers:

"faineant.bas"

"femme.seq"

seraient ainsi copiés. Il existe cependant un autre joker, le point d'interrogation. Un point d'interrogation remplace, en n'importe quelle position, une lettre.

On aurait donc pu également écrire au lieu de filecopy f*.*, filecopy f???????.??? ou filecopy f*.??? mais aussi filecopy f???????.*. Si vous avez par exemple sauvegardé un agenda sur disquette, vous pourriez avec l'instruction:

filecopy 05-??.*

faire copier tous les rendez-vous du mois de mai pour les années précédentes, à condition bien entendu que le format du nom de fichier corresponde à ce modèle.
filecopy 1?1 copierait tous les fichiers qui ont un 1 en première et en troisième positions, alors que tous les caractères autorisés seraient imaginables pour l'emplacement du milieu.

Lorsque vous utilisez l'instruction filecopy *.*, on vous demande si vous voulez copier tous les fichiers ou si vous voulez effectuer une sélection. On vous montre les noms de fichier les uns après les autres et vous pouvez décider chaque fois (avec y pour oui et n pour non) si le fichier doit être copié.

1.3.11 L'INSTRUCTION DIR

Avec l'instruction DIR vous pouvez faire éditer le catalogue de la disquette. Vous pouvez d'autre part effectuer une sélection, c'est-à-dire que vous pouvez exclure certains fichiers. Ceci est obtenu grâce à l'utilisation des jokers. Si vous négligez les paramètres, on suppose que vous avez entré *.*. L'instruction DIR a les effets suivants:

DIR	:affiche tous les fichiers
DIR B:	:affiche tous les fichiers du lecteur B:
DIR *.BAS	:affiche uniquement les fichiers BASic
DIR FILECOPY.COM	:affiche uniquement le fichier FILECOPY.COM, pour autant qu'il soit présent sur la disquette

Les fichiers sont affichés dans l'ordre dans lequel ils ont été créés, c'est-à-dire que le premier fichier que l'on a sauvegardé sur la disquette sera exactement le premier présenté dans le catalogue.

1.3.12 L'INSTRUCTION ERA

Avec l'instruction ERA (pour ERAse), vous pouvez supprimer des fichiers sur la disquette. Ce ne sont toutefois que les entrées correspondant à ces fichiers sur la disquette qui sont supprimées, les données elles-mêmes ne sont pas détruites mais il n'est plus possible d'y accéder. Si vous indiquez *.*, l'instruction doit être confirmée car toutes les entrées de la disquette seraient ainsi supprimées. Si un fichier à supprimer ainsi ne peut qu'être lu (voir aussi 1.3.16), l'exécution de

l'instruction sera interrompue.

Instructions possibles:

ERA DISCCOPY.COM :Supprime le fichier DISCCOPY.COM
ERA *.SEQ :Supprime tous les fichiers marqués .SEQ

1.3.13 L'INSTRUCTION REN

L'instruction REN (pour REName) permet de changer le nom de fichiers.

REN nouveau nom=ancien nom

Si le nouveau nom existe déjà ou si l'ancien nom n'existe pas, un message d'erreur est sorti.

1.3.14 L'INSTRUCTION TYPE

L'instruction TYPE permet de faire éditer le contenu de fichiers sur l'écran. S'il ne s'agit pas de fichiers ASCII, comme par exemple pour les programmes Basic, des symboles graphiques ou autres peuvent apparaître sur l'écran. Il se peut également par exemple que la couleur du fond ou que le mode d'affichage soient modifiés.

TYPE EX1.BAS :Affichage du programme d'exemple EX1

1.3.15 COMMUTATION DU LECTEUR DE DISQUETTE STANDARD

Les instructions A: et B: vous permettent, si vous possédez deux lecteurs de disquette, de commuter d'un lecteur de disquette à l'autre.

B: le lecteur de disquette B: est le lecteur standard

Le symbole d'interrogation est également modifié pour devenir B> par exemple.

1.3.16 L'INSTRUCTION PIP

L'instruction PIP vous permet d'effectuer un transfert entre l'ordinateur et la périphérie, à moins que vous n'utilisiez cette instruction pour effectuer une copie si vous possédez deux lecteurs de disquette.

La syntaxe est:

PIP<objet>=<source>.

La <source> et l'<objet> peuvent être des fichiers ou des périphériques. Vous pouvez appeler les périphériques suivants:

Comme source:

CON: Console = clavier
RDR: Interface sérielle

Comme objet:

CON: Console = écran
PUN: Interface sérielle
LST: Imprimante

Exemple:

Vous voulez créer un fichier que vous voulez entrer à partir du clavier:

PIP Text.Txt=CON:

Tout ce que vous entrez sera alors sauvegardé sur le fichier Text.Txt, jusqu'à ce que vous appuyiez sur la touche [CTRL]-Z. Le fichier sera alors fermé.

Vous ne pouvez pas utiliser l'instruction PIP pour copier des fichiers avec un seul lecteur de disquette. Utilisez dans ce cas l'instruction FILECOPY.

Autres exemples:

PIP LST:=EX2.BAS   Sort le fichier Basic EX2.BAS sur imprimante

PIP CON:=EX2.BAS Sort le fichier Basic EX2.BAS sur écran. Cette instruction est semblable à l'instruction TYPE EX2.BAS

1.3.17 L'INSTRUCTION STAT

STAT est mis pour STATus (statut). L'instruction STAT vous permet d'une part de réaliser une sortie pratique des informations sur vos fichiers, de façon semblable à l'instruction DIR.

STAT
STAT B:
STAT *.BAS

sont des exemples possibles. D'autre part cette instruction vous offre une option très importante et très pratique. Vous pouvez rendre par exemple un fichier "uniquement lisible", c'est-à-dire que vous ne pourrez plus effacer ce fichier par mégarde. Avec l'instruction:

STAT *.BAS$R/O

tous les fichiers Basic seront dotés du statut Read-Only. Tout effaçage par mégarde de ces fichiers est dès lors exclu.

Pour supprimer ce statut, il vous faut ajouter les caractères pour le statut de lecture/écriture ($R/W) à l'instruction de base:

STAT *.BAS$R/W

Vous pouvez également doter un fichier du statut système. Ce fichier ne sera plus alors listé par l'instruction DIR, il deviendra en fait invisible. Ce fichier ne pourra d'autre part plus être copié. Il ne sera plus listé que par l'instruction STAT. Si vous entrez par exemple:

STAT *.COM$SYS

tous les fichiers COM seront dotés du statut système, vous ne pourrez donc plus lister ces fichiers avec l'instruction DIR et vous ne pourrez plus les copier. Toutefois ces instructions peuvent continuer d'être appelées.

L'inverse de cette instruction est:

STAT *.COM$DIR

Cette instruction fixe le "statut catalogue".

Voici des instructions STAT impressionnantes:

STAT dsk: Toutes les informations importantes sur le format de la disquette vous sont fournies.

STAT val: Vous fournit une liste des abréviations et vous indique quelle est leur affectation actuelle. Par exemple CON:=TTY etc...

ou STAT dev: et STAT usr:.

1.3.18 BOOTGEN

BOOTGEN permet de copier les deux pistes 0 et 1 sur une autre disquette. Vous pouvez utiliser cette instruction lorsque vous voulez doter de CP/M une disquette formatée en format Vendor ou lorsque vous voulez copier un nouveau secteur de configuration sur plusieurs disquettes.

1.3.19 MOVCPM et SYSGEN

L'instruction MOVCPM vous permet de décaler CP/M dans une autre zone de la mémoire. Ceci est souvent nécessaire lorsque CP/M et un logiciel quelconque se chevauchent. Vous pouvez décaler CP/M par pas de 256 octets. La syntaxe est:

MOVCPM<grandeur>*

La grandeur a des valeurs comprises entre 64 et 179. Le CP/M standard a été produit avec la grandeur 179. MOVCPM décale par exemple CP/M de 256 octets vers le bas.

Vous pouvez ensuite sauvegarder le CP/M nouvellement produit avec l'instruction SYSGEN ou le sauvegarder dans un fichier. Cette possibilité vous est également donnée sous MOVCPM.

SYSGEN écrit le résultat d'une instruction MOVCPM sur la piste système. Il y cependant encore trois instructions différentes à cet effet:

SYSGEN*

Cette instruction écrit le CP/M créé immédiatement auparavant par une instruction MOVCPM sur les pistes système.

SYSGEN<nom de fichier>

Cette instruction lit le fichier créé par MOVCPM sous le nom <nom de fichier> et le copie sur les pistes système. Exemple: SYSGEN CPMEXTRA.COM

SYSGEN

A défaut de paramètres, on vous demande quelles sont les disquettes source et objet et CP/M sera copié en fonction de cela sur la disquette objet. Cette instruction vous permet de copier CP/M sur une disquette VENDOR.

1.3.20 SETUP

Cette instruction vous permet de modifier de façon décisive l'allure et le fonctionnement de CP/M. Cette instruction vous permet d'intervenir dans l'affectation des touches, dans l'accès à la disquette et dans beaucoup d'autres choses. Vous ne devriez cependant utiliser cette instruction qu'après vous être informé de façon suffisamment précise sur les effets qu'elle peut avoir. Les chapitres 3.7.3.2 et 1.5 du manuel de l'utilisateur du lecteur de disquette DDI-1 fournissent notamment les informations nécessaires à cet égard.

1.3.21 AMSDOS

L'instruction AMSDOS déconnecte CP/M et vous ramène à AMSDOS. Vous pouvez alors programmer en Basic comme à l'habitude. Vous disposez des instructions AMSDOS.

1.4 LE TRAVAIL SOUS AMSDOS

1.4.1 QU'EST-CE QU'AMSDOS?

AMSDOS signifie "AMStrad Disc Operating System". AMSDOS soutient le travail avec la disquette en Basic. Respectez absolument le bon ordre pour la mise sous tension! C'est d'abord le lecteur de disquette qui doit être mis sous tension et ce n'est qu'ensuite que vous pouvez allumer le moniteur et l'ordinateur. La raison en est que lors de la mise sous tension de l'ordinateur un test est automatiquement effectué pour savoir quels éléments de la périphérie (imprimante, lecteur de disquette, etc...) sont allumés. Si donc le lecteur de disquette est encore hors tension lorsque vous allumez l'ordinateur, les instructions qui devraient être envoyées plus tard au lecteur de disquette seront malgré tout envoyées au lecteur de cassette.

Toutes les instructions ci-dessous, qui sont normalement envoyées au lecteur de cassette, sont envoyées au lecteur de disquette si vous ne donnez pas d'instruction contraire:

```
load"nom de fichier"
run"nom de fichier"
chain"nom de fichier"
merge"nom de fichier"
chain merge"nom de fichier"
save"nom de fichier"
openin"nom de fichier"
closein
openout"nom de fichier"
closeout
cat
eof
input#9
line input#9
print#9
write#9
list#9
```

L'instruction speed write se rapporte cependant toujours au lecteur de cassette car sur le lecteur de disquette, les vitesses de lecture et d'écriture (voir chapitre 1) ne sont pas variables.

Il existe en outre encore, EN PLUS, les instructions suivantes, appelées instructions externes. Il s'agit ici d'instructions externes car ces instructions sont stockées dans la ROM de l'AMSDOS, c'est-à-dire dans le lecteur de disquette. Ces instructions ne peuvent absolument pas être utilisées en Basic cassette car elles ne deviennent disponibles qu'après la mise sous tension du lecteur de disquette. Ces instructions commencent par un | (SHIFT et); ce caractère introduit toutes les instructions dites instructions RSX, c'est-à-dire les instructions d'extension du Basic. Les instructions disquette ne sont toutefois pas des instructions RSX à proprement parler mais constituent une exception.

|a
|b
|dir
|disc
|disc.in
|disc.out
|drive
|era
|ren
|tape
|tape.in
|tape.out
|user

Ces instructions seront expliquées encore une fois par des exemples spécifiques.

1.4.2 LOAD ET RUN

Lorsque vous voulez charger un programme à partir de la disquette, que ce soit pour le lancer ou pour programmer, vous devez utiliser l'instruction:

LOAD"nom de fichier"

Cette instruction vous permet de charger des programmes Basic. Lorsque vous n'indiquez pas le type de fichier, AMSDOS suppose ".   ", c'est-à-dire qu'aucun caractère ne figure dans le type de fichier.

AMSDOS essaie d'abord de charger le fichier "nom de fichier.   ". Si ce

fichier n'existe pas, il essaie de charger le fichier "nom du fichier.BAS". Si ce fichier n'existe pas non plus, il essaie enfin de charger le fichier "nom de fichier.BIN". Ce n'est que si ce fichier n'a pas non plus été trouvé qu'un "File not found" est sorti. Vous pouvez cependant entrer aussi:

LOAD"nom de fichier.BAS"

pour empêcher qu'AMSDOS ne charge un éventuel fichier qui existerait sous le nom de "nom de fichier.   ".

Il est bien sûr possible de charger des programmes à partir du lecteur de disquette B:. Il vous suffit pour cela d'indiquer le lecteur de disquette dans l'instruction LOAD:

LOAD"B:nom de fichier"

Si aucune disquette ne se trouve dans le lecteur ou si elle n'y pas été placée correctement, le message d'erreur suivant apparaît:

Drive A:disc missing
Retry, Ignore or Cancel?

Placez alors la disquette correctement dans le lecteur et actionnez la touche "r" pour "Retry", c'est-à-dire réessayer.

Notez qu'il n'est pas nécessaire d'indiquer le type de fichier. AMSDOS ajoute automatiquement au nom de fichier le suffixe ".BAS" lors du chargement et de la sauvegarde des programmes Basic. Si vous avez toutefois sauvegardé un programme sous un autre nom avec une autre désignation que ".BAS", vous pouvez également charger ce fichier avec l'instruction:

LOAD"nom de fichier.xxx"

Vous pouvez remplacer ici xxx par la marque de type de fichier correspondante.

Si toutefois le fichier "nom de fichier" n'existe pas (par exemple parce que vous avez mal entré le nom de fichier), vous obtenez le message d'erreur:

Nom de fichier.   not found.

Vérifiez alors votre nom de fichier et essayez encore une fois si nécessaire.

Si vous obtenez le message d'erreur Type mismatch, cela signifie que vous avez oublié d'entrer les guillemets au début du nom de fichier.

Le message d'erreur:

Drive A:read fail
Retry, Ignore or Cancel

indique qu'une erreur de lecture s'est produite sur la disquette. C'est probablement que votre disquette est défectueuse (ou que vous n'avez pas placé la bonne disquette dans le lecteur). Il se peut également que votre disquette n'ait pas été formatée correctement dans le format Amstrad.

Le message:

Press PLAY then any key:

signifie que la connexion entre l'ordinateur et l'interface ou entre l'ordinateur et le lecteur de disquette n'est pas bonne. Il se peut également que vous n'ayez pas allumé le lecteur de disquette en premier. Une autre possibilité est que vous ayez utilisé l'instruction |tape.in, entrez alors:

|disc

et essayez à nouveau. Si vous obtenez le même message d'erreur, éteignez votre ordinateur et allumez-le à nouveau.

Comme en Basic cassette, il est possible de faire démarrer les programmes automatiquement après leur chargement. On utilise également à cet effet l'instruction:

RUN"nom de fichier"

Il n'y a cependant plus comme en Basic cassette la possibilité d'utiliser l'instruction RUN" car vous devez définir précisément le nom du fichier à charger. Les mêmes messages d'erreur expliqués pour LOAD, avec les mêmes

conséquences, valent pour l'instruction RUN"nom de fichier".

Vous remarquerez notamment lors du chargement des programmes Basic combien votre lecteur de disquette est rapide par rapport au lecteur de cassette.

Vous pouvez tout aussi naturellement utiliser également les autres instructions Basic de chargement. Pour l'utilisation des instructions:

CHAIN"nom de fichier", MERGE"nom de fichier" et CHAIN MERGE"nom de fichier"

nous vous invitons à vous reporter au manuel d'utilisation.

Une très utile possibilité est constituée par le fait que vous pouvez sous AMSDOS, comme sous CP/M, définir des Users. Si vous avez déjà examiné un catalogue, vous avez certainement constaté que la première ligne est:

Drive A:user 0

Vous pouvez définir des numéros user de 0 à 15, le numéro user standard est 0. Par l'intermédiaire des numéros user, vous pouvez appeler différents catalogues, c'est-à-dire que vous pouvez par exemple placer sous user 0 uniquement les instructions CP/M et sous user 1 vos programmes Basic et les données. Si vous voulez charger un programme de user 1, vous avez deux possibilités pour le faire. La première consiste à définir le user avec l'instruction 1USER:

IUSER,1

vous permet de le faire. Vous pouvez alors charger le programme avec LOAD"nom de fichier". Après chargement du programme, vous vous trouvez encore dans user 1, jusqu'à ce que vous le redéfinissiez. La seconde possibilité consiste à indiquer le user dans le nom de fichier:

LOAD"1:nom de fichier"

chargera un programme du user 1, quel que soit votre statut user actuel. Vous devez donc indiquer le numéro user, suivi d'un double-point puis du nom de fichier, de même que pour le choix du lecteur de disquette. Si vous voulez sélectionner simultanément le lecteur de disquette et le

numéro user, il vous faut indiquer d'abord le numéro user, ensuite le lecteur de disquette et placer enfin le double-point. Exemple:

LOAD"8B:Nim"

chargera le jeu "Nim" à partir du lecteur B, user étant 8. La syntaxe est donc:

LOAD"<NoUser><Drive>:Nom de fichier"

<NoUser> est un numéro user quelconque entre 0 et 15, <Drive> est mis pour le nom du lecteur de disquette sélectionné, donc soit A, soit B.

1.4.3 LES INSTRUCTIONS FICHIER

Pour traiter des fichiers avec le lecteur de cassette, on a besoin des instructions:

```
OPENOUT"nom de fichier"
OPENIN"nom de fichier"
CLOSEOUT
CLOSEIN
INPUT#9
LINE INPUT#9
PRINT #9
WRITE #9
```

Le programme suivant écrit les 100 premiers nombres sur un fichier sur cassette:

```
10 REM ======================================
20 REM Programme exemple de fichiers données
30 REM ======================================
40:
50 OPENOUT"nombres"
60 FOR I=1 TO 100
70 PRINT#9,I
80 NEXT I
90 CLOSEOUT
100:
110 REM ======================================
```

```
120 REM Maintenant lire
130 REM =====================================
140:
150 MODE 1:PEN 1
160 PRINT"Rembobinez la bande"
170 PRINT"et frappez une touche"
180 D$=INKEY$:REM *** Vide le buffer ***
190 IF INKEY$="" THEN 190
200:
210 OPENIN"nombres"
220 WHILE NOT EOF
230 INPUT V9,I
240 PRINT I,
250 WEND
260 CLOSEIN
270 END
```

Attention! Ce programme n'écrira sur la cassette que si vous n'avez pas connecté le lecteur de disquette à l'ordinateur. Si vous avez donc connecté votre lecteur de disquette à l'ordinateur, entrez encore l'instruction:

ITAPE

avant de faire exécuter le programme. Maintenant toutes les instructions fichiers vont à nouveau sur le lecteur de cassette. Vous pouvez ainsi mesurer le temps que le programme met à être exécuté. Lorsque ce sera terminé, entrez:

Idisc

et lancez à nouveau le programme. Maintenant, le fichier "nombres" est ouvert sur le lecteur de disquette. Comme vous ne pouvez pas rembobiner une disquette, vous pouvez ignorer le message "Rembobinez la bande" et frapper directement une touche.

Placez maintenant une disquette non protégée contre l'écriture dans le lecteur, de préférence une disquette que vous avez formatée pour y faire des expériences. Entrez ensuite RUN.

Vous remarquerez que ce travail est exécuté beaucoup plus rapidement. En principe vous n'avez donc pas à changer votre façon de travailler si vous

voulez traiter des fichiers séquentiels sur disquette et que vous l'aviez déjà fait en Basic avec le lecteur de cassette.
Vous pouvez également revenir avec l'instruction ltape au Basic cassette bien connu. Nous en dirons cependant plus à ce sujet au chapitre 1.4.4.

La fonction eof, telle qu'elle se présente dans la ligne 220 de notre programme d'exemple signifie End Of File, soit fin du fichier. Cette fonction est fausse (=0) si d'autres caractères peuvent encore être lus à partir du fichier ouvert en lecture. EOF devient vrai (=-1) si le dernier caractère du fichier a été lu et qu'il n'est donc plus possible de retirer d'autres caractères de ce fichier. Cette instruction est très importante lorsqu'on lit des fichiers séquentiels et que le nombre de caractères à lire n'est pas connu d'avance. EOF vous fournit ainsi une possibilité de travailler de façon très souple avec les fichiers séquentiels.

Nous nous occupons ici exclusivement des instructions disquette! Les effets sur le travail avec le lecteur de cassette sont comparables mais ils n'ont pas à être décrits en détail dans un ouvrage consacré au lecteur de disquette.

Entrez maintenant l'instruction:

cat

Vous obtenez le catalogue de la disquette sur l'écran. On entend par catalogue de la disquette la liste de tous les fichiers sauvegardés sur une face de disquette. La disquette doit être organisée; il existe à cet effet des blocs spéciaux sur la disquette sur lesquels figurent les informations suivantes: le nom des différents fichiers, leur taille, le type auquel ils appartiennent et où le DOS, le Disk Operating System, peut trouver ces fichiers.

Si vous examinez avec l'instruction cat ou avec Idir le catalogue de la disquette, vous obtenez des informations sur les fichiers figurant sur la disquette et sur la place encore disponible pour des sauvegardes sur la disquette.

Si vous utilisez l'instruction cat, on vous indique en outre encore la longueur des différents fichiers en Koctets, ce qui manque avec l'instruction Idir. La plus petite unité pour un fichier est le Koctets.

Les instructions Idir et cat ne sont pas identiques. Toutes deux affichent le catalogue de la disquette mais l'instruction Idir affiche les fichiers dans l'ordre dans lequel ils ont été placés sur la disquette alors que l'instruction cat affiche la taille de chaque fichier et trie en outre les entrées du catalogue par ordre alphabétique avant de les sortir sur l'écran. Il y a donc là deux différences importantes.

Vous allez trouver également dans ce catalogue de la disquette le fichier "nombres" que nous venons de créer:

nombres .   1k

Comme nous n'avons pas indiqué de type de fichier, aucun type de fichier n'est indiqué après le point. Les noms de fichiers sont toujours complétés par des espaces si nécessaire, de façon à ce qu'ils comportent toujours exactement 8 caractères.

Vous voyez que nos 100 nombres occupent 1 Koctets, soit 1024 octets. Mais même si nous n'avions écrit qu'un nombre dans ce fichier, le fichier "nombres" occuperait de toute façon 1 Koctets puisque c'est la plus petite unité possible.

Mais lancez maintenant votre programme à nouveau et voyez ce que devient le catalogue. Entrez RUN, puis cat.

Vous verrez qu'il y a maintenant deux fichiers portant le nom "nombres". Un des fichiers ne comporte pas d'indication du type de fichier. C'est dans ce fichier que figurent les 100 nombres que vous avez sauvegardés en dernier. Il y a cependant maintenant encore un fichier "nombres .BAK". Vous vous souvenez certainement, comme nous vous l'expliquions dans la partie consacrée à CP/M, que BAK est l'abréviation de BAcKup. Lors de l'ouverture du fichier "nombres" avec l'instruction OPENOUT"nombres", AMSDOS a testé si le fichier "nombres" existait déjà. Comme nous avions déjà fait exécuter notre programme une fois, ce fichier existait déjà. C'est d'ailleurs ce que nous avons pu constater d'après le catalogue. Comme AMSDOS ne veut pas effacer ce fichier, il en réalise d'abord une copie de sécurité sous le nom de "nom de fichier.BAK". C'est alors seulement que l'ancien fichier est véritablement supprimé, de façon à ce que le nouveau fichier puisse être ouvert sous le même nom. Vous apprécierez certainement à l'usage cette particularité d'AMSDOS. Si vous êtes certain de ne plus avoir besoin de la copie de sécurité, vous pouvez également la supprimer.

Lorsque vous sauvegardez un programme Basic avec l'instruction SAVE"nom de fichier" ou SAVE"nom de fichier.BAS",A, un programme pouvant déjà figurer sur la disquette sous le même nom sera également d'abord sauvegardé sous forme de fichier Backup, de sorte que vos programmes Basic sont protégés de façon assez efficace contre un effaçage inopiné. Essayez donc ce qui suit:

Entrez NEW.

NEW (ENTER)

```
10 PRINT"Premiere partie."
20 PRINT
```

SAVE "Prog"

Le programme est sauvegardé sur la disquette sous le nom de "Prog   .BAS". Ajoutez encore les lignes:

```
30 PRINT"a ete complete.
40 PRINT
```

Entrez à nouveau l'instruction de sauvegarde d'un programme:

SAVE"Prog"

Vous utilisez donc exactement le même nom de fichier qu'auparavant. Vous pouvez maintenant entrer LOAD"Prog.BAK". (Notez que lorsque vous entrez un nom de fichier il n'est pas obligatoire qu'il y ait effectivement 8 caractères avant le point.) Si vous faites maintenant lister votre programme, vous constatez que c'est la première version de notre petit programme que vous avez chargée. Entrez maintenant:

LOAD"Prog"

et c'est la dernière version avec 4 lignes de programme qui sera chargée en mémoire. Supposons que vous complétiez encore le programme en y ajoutant la ligne:

```
50 END
```

et que vous le sauvegardiez une nouvelle fois sur la disquette, la toute

première version du programme qui ne comportait que deux lignes aura cette fois disparu. Vous trouvez maintenant notre programme de 4 lignes sous le nom de "Prog.BAK" et le programme actuel a pour nom "Prog.BAS".

Si vous sauvegardiez maintenant le programme sous le nom de "nombres", aucun fichier de copie ne serait réalisé car il n'existe aucun fichier sous le nom de "nombres.BAS". Vos fichiers "nombres" et "nombres.BAK" seront donc conservés. Si toutefois vous sauvegardez le programme encore une fois, le fichier "nombres.BAK" sera effacé et remplacé par une copie du programme Basic.

Voici les autres instructions SAVE possibles:

SAVE"nom de fichier",A

sauvegarde le programme sous la forme d'un fichier ASCII. Ces fichiers de programme sont un peu plus longs que les fichiers de programme Basic "normaux" car les différentes instructions comme "PRINT" par exemple sont sauvegardées sur la disquette caractère par caractère et non sous la forme d'un token. (Un token est un code d'un octet représentant un mot-clé Basic.) Les programmes qui ont été sauvegardés avec cette instruction peuvent également être lus caractère par caractère par un programme.

SAVE"nom de fichier",P

sauvegarde un programme protégé, c'est-à-dire qu'on ne peut plus LISTer le programme et qu'après une interruption par (ESC)(ESC) il n'est plus possible de le relancer ni de le sauvegarder.

SAVE"nom de fichier",B,<adresse de départ>,<longueur>

Cette instruction vous permet de sauvegarder une zone de la mémoire, par exemple la zone de la mémoire-écran, sous la forme d'un fichier binaire. Vous devez indiquer l'adresse de départ et la longueur de la zone de mémoire. Lors du chargement avec l'instruction LOAD"nom de fichier.BIN", le fichier sera alors à nouveau chargé dans l'adresse originale correspondante.

Vous pouvez bien sûr fournir également dans le nom de fichier pour l'instruction SAVE, exactement comme pour l'instruction LOAD, des indications sur le lecteur de disquette et sur le numéro user.

Vous avez certainement remarqué que vous pouvez travailler avec l'Instruction SAVE comme vous y étiez habitué avec le Basic cassette. L'utilisation des autres instructions Basic de chargement et d'écriture de données vous sera expliquée dans le chapitre suivant, où nous vous expliquerons également, par des exemples, le travail avec des fichiers séquentiels.

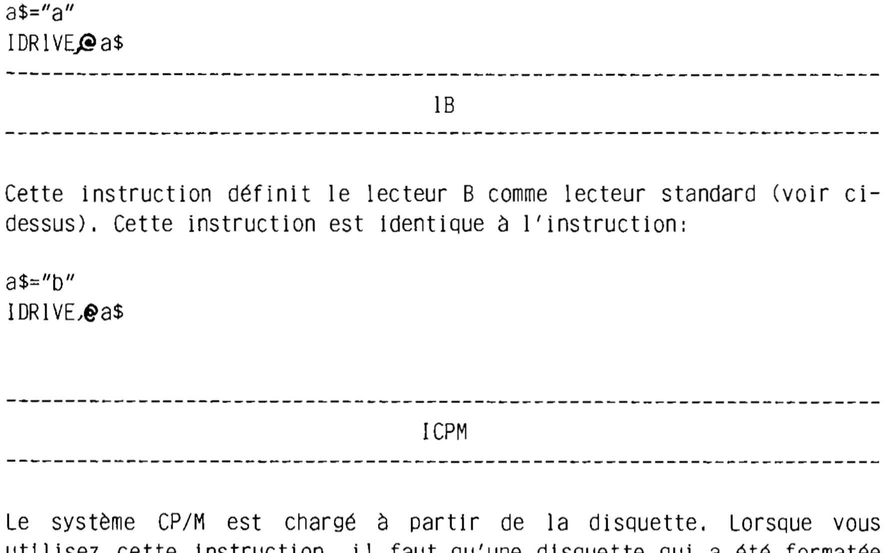
1.4.4 LES INSTRUCTIONS SUPPLEMENTAIRES AMSDOS

En connectant le lecteur de disquette, vous avez à votre disposition encore d'autres instructions disquette qui sont sauvegardées dans la ROM (Read Only Memory) du lecteur de disquette. Ces instructions sont également appelées instructions externes parce qu'elles sont définies dans le lecteur de disquette.
Sans lecteur de disquette, vous ne pouvez utiliser ces instructions. Vous pouvez identifier ces instructions par le fait qu'elles commencent par le caractère |(Touches SHIFT et @). Vous pouvez aussi bien entrer ces instructions directement que les intégrer dans des programmes. Voici la liste des instructions externes avec des exemples caractéristiques.

Lorsqu'un mot est entouré par les caractères < et >, cela signifie qu'il s'agit d'un qualificatif de paramètre. Par exemple, vous trouverez souvent l'expression <expression alphanumérique> ce qui signifie que vous devez insérer à cet endroit une variable alphanumérique dans l'instruction. Mais si l' <expression alphanumérique> est placée entre crochets [<expression alphanumérique>], cela vous indique que l' <expression alphanumérique> peut être également négligée et qu'elle est donc facultative.

|A

Cette instruction définit le lecteur A comme lecteur standard. Cette instruction est bien sûr inutile si vous n'avez qu'un lecteur de disquette puisque ce lecteur de disquette A est automatiquement le lecteur standard. On entend par lecteur standard le lecteur de disquette qui est automatiquement appelé par les instructions disquette lorsque le lecteur de disquette n'est pas précisé dans l'instruction. Cette instruction est identique à l'instruction:



a$="a"
IDRIVE,@a$

IB

Cette instruction définit le lecteur B comme lecteur standard (voir ci-dessus). Cette instruction est identique à l'instruction:

a$="b"
IDRIVE,@a$

ICPM

Le système CP/M est chargé à partir de la disquette. Lorsque vous utilisez cette instruction, il faut qu'une disquette qui a été formatée avec CP/M sur les deux pistes système se trouve dans le lecteur A. Il peut s'agir par exemple de la disquette fournie avec le lecteur de disquette.

IDIR[,<expression alphanumérique>]

Cette instruction vous montre le contenu de la disquette. Elle est très comparable à l'instruction Basic standard CAT.
On vous montre le catalogue de la disquette ainsi que la place encore disponible sur la disquette.
Si l' <expression alphanumérique> facultative manque, on supposera *.*, ce qui veut dire que tous les fichiers seront sortis (sans sélection). Comme nous l'avons déjà expliqué, les étoiles et les points d'interrogation sont des jokers qui représentent n'importe quels caractères. En guise de rappel: un point d'interrogation signifie: ici peut se trouver n'importe quel caractère; une étoile signifie: à partir d'ici et jusqu'à la fin du nom de fichier peuvent figurer n'importe quels caractères. Supposons que vous vouliez avoir une liste uniquement des fichiers Basic, vous l'obtiendrez avec l'instruction:

```
a$="*.BAS"
|DIR,@a$
```

L'utilisation des jokers peut être très utile et vous faire gagner beaucoup de temps. Essayez de retenir l'utilisation de ces caractères de façon à ce que vous puissiez les mettre vous-même en oeuvre. Il arrive souvent en effet qu'on ne veuille par exemple avoir une liste que des fichiers Basic ou que des fichiers séquentiels. Avec l'instruction |DIR, la sélection que vous pouvez ainsi effectuer vous permet d'obtenir une plus grande clarté des informations.

L'avantage de l'instruction CAT est qu'elle trie les noms de fichier avant de les sortir et qu'elle indique la taille des fichiers. L'instruction |DIR a l'avantage de permettre un affichage SELECTIF du catalogue.

|DISC

Cette instruction comprend les deux instructions |disc.in et |disc.out. Comme vous devez utiliser les mêmes instructions Basic pour le lecteur de cassette et pour le lecteur de disquette, vous ne pourriez en principe appeler avec succès qu'un seul des deux périphériques. Mais pour que vous puissiez également utiliser le lecteur de cassette lorsque vous travaillez avec le lecteur de disquette, les instructions |disc et |tape vous permettent de sélectionner librement le périphérique d'entrée ou de sortie.

|DISC.IN

Après que cette instruction ait été entrée, les instructions d'ENTREE suivantes se rapporteront toutes automatiquement au lecteur de disquette, et non au lecteur de cassette:

```
LOAD"Nom de fichier"
RUN"Nom de fichier"
CHAIN"Nom de fichier"
```

```
CHAIN MERGE"Nom de fichier"
MERGE"Nom de fichier"
OPENIN"Nom de fichier"
CLOSEIN
EOF
CAT
INPUT#9
LINE INPUT#9
```

Cette instruction déconnecte à nouveau l'instruction ITAPE ou ITAPE.IN. Si vous avez par exemple la séquence d'instructions suivante:

```
ITAPE
IDISC.IN
OPENIN"Fichier"
OPENOUT"Backup"
```

un canal d'entrée sera ouvert en lecture sur le lecteur de disquette mais le canal de sortie écrira sur la cassette. C'est ainsi que vous pouvez par exemple copier des fichiers de la disquette sur la cassette (pour des raisons de sécurité des données).

IDISC.OUT

Cette instruction est comparable à l'instruction IDISC.IN mais ce sont ici les sorties qui sont envoyées sur le lecteur de disquette. Les instructions suivantes sont concernées:

```
SAVE"Nom de fichier"
OPENOUT"Nom de fichier"
CLOSEOUT
WRITE#9
PRINT#9
```

```
ITAPE
IDISC.OUT
IOPENIN"Backup"
IOPENOUT"Fichier"
```

C'est donc l'inverse de l'exemple donné pour |DISC.IN. Maintenant le fichier Backup sera lu à partir de la cassette avec les instructions de lecture mais les instructions d'écriture se rapporteront au lecteur de disquette.

|DRIVE,<expression alphanumérique>

Cette instruction est identique aux instructions |A et |B mais cette instruction est plus souple car le nom du lecteur de disquette standard à sélectionner peut être fourni sous forme d'une variable alphanumérique.

```
a$="b"
|DRIVE,@a$
```

aura pour conséquence que le lecteur de disquette B sera défini comme lecteur standard à partir du moment où cette instruction aura été exécutée. Vous ne pouvez bien sûr pas utiliser cette instruction si vous n'avez qu'un lecteur de disquette.

|ERA,<expression alphanumérique>

ERA est mis pour le mot anglais ERAse=supprimer. Cette instruction vous permet donc de supprimer des fichiers. Vous pouvez également utiliser des jokers avec cette instruction. Par exemple, l'instruction:

```
a$="*.seq"
|ERA,@a$
```

supprimera tous les fichiers ayant le type de fichier "seq". Il convient cependant de toujours bien réfléchir avant d'utiliser cette instruction, notamment lorsqu'on travaille avec des jokers. Il peut en effet arriver dans ce dernier cas que vous supprimiez également des fichiers que vous n'aviez pas l'intention de supprimer. Il est donc déconseillé d'utiliser des jokers avec l'instruction |ERA si l'on n'a pas encore l'habitude du maniement du lecteur de disquette et de l'utilisation des jokers.

L'instruction:

```
a$="*.*"
|ERA,@a$
```

supprimera tous les fichiers. Dans ce cas AMSDOS attend une confirmation spéciale de l'instruction.

|REN,<expression alphanumérique>,<expression alphanumérique>

REName= changement du nom d'un fichier. Cette instruction vous permet donc de donner un autre nom à un fichier. La première expression alphanumérique doit contenir le NOUVEAU nom de fichier, la seconde contient l'ancien nom de fichier. Si vous voulez par exemple changer le nom du fichier "Pepe" parce que vous préférez votre mémé, vous pouvez entrer les instructions suivantes:

```
ancien$="Pepe.bas"
nouveau$="Meme.bas"
|REN, nouveau$,ancien$
```

Si vous examinez maintenant le catalogue avec l'instruction |DIR, vous verrez que le fichier "Pepe.BAS" n'existe plus et qu'il y a par contre maintenant un fichier "Meme.BAS".
L'indication des deux expressions alphanumériques est obligatoire.

|TAPE

Cette instruction annule les instructions |DISC, |DISC.IN et |DISC.OUT et place l'entrée comme la sortie sur le lecteur de cassette. Cette instruction comprend les fonctions des instructions |TAPE.IN et |TAPE.OUT.

ITAPE.IN

Cette instruction fait que le lecteur de cassette sera utilisé en entrée. Cette instruction annule les instructions IDISC et IDISC.IN.

ITAPE.OUT

Le lecteur de cassette sera utilisé comme périphérique de sortie. Cette instruction annule les instructions IDISC et IDISC.OUT.

IUSER,<expression entière>

Cette instruction vous permet de définir un user = utilisateur déterminé. Vous avez certainement déjà remarqué que dans les catalogues de la disquette apparaît également le message:

USER:0

Il s'agit d'une très utile instruction spéciale CP/M. Vous trouverez des informations plus complètes à ce sujet au chapitre 1.4.2.

L'instruction IUSER vous permet par exemple de protéger vos fichiers contre les regards indiscrets.
Sauvegardez donc un programme Basic quelconque de la façon suivante:

```
IUSER,3
SAVE"Programme"
IUSER,0
CAT
```

ou beaucoup plus simplement avec l'instruction:

```
SAVE"3:Programme"
```

Vous ne trouverez pas le programme que vous venez de sauvegarder dans le catalogue. Mais si vous entrez:

```
|USER,3
CAT
|USER,0
```

vous verrez s'afficher un catalogue beaucoup plus court puisqu'il ne comporte qu'une entrée. Vous pouvez donc constituer plusieurs catalogues différents ou dissimuler certains programmes à d'autres utilisateurs.

Vous avez certainement remarqué que les instructions avec des expressions alphanumériques sont particulièrement difficiles à utiliser. Vous êtes en effet obligé d'entrer par exemple:

```
a$="a"
|DRIVE,@a$
```

alors qu'il serait beaucoup plus simple d'entrer:

```
|DRIVE,"a"
```

Cependant le système d'exploitation Basic prévoit que l'adresse d'une expression alphanumérique doit être transmise au ~~système~~ d'exploitation AMSDOS. AMSDOS va alors lui-même chercher cette chaîne de caractères dans la mémoire.

Les chaînes de caractères peuvent être placées n'importe où dans la mémoire. La fonction @nom de variable vous permet de faire afficher l'adresse à laquelle est stockée la variable, en l'occurrence une chaîne de caractères. Cette forme de transmission des données n'est certainement pas très pratique mais vous vous habituerez vite à cette procédure.
L'instruction @ est une instruction Basic utile qui n'est pas évoquée dans le manuel d'utilisation. Entrez par exemple:

```
a=12.2
PRINT@a
```

Suivant le nombre de variables que vous avez déjà utilisées, la valeur sortie en réponse sera plus ou moins élevée. Plus une variable est déclarée tardivement, plus grande sera la valeur de la fonction nom de variable. La longueur du programme Basic actuel joue par ailleurs

également un rôle car la table des variables commence dans la mémoire immédiatement après le programme Basic.

Comme vous le savez peut-être, les variables numériques sont placées dans la mémoire de bas en haut alors que les chaînes sont situées contre la limite supérieure de la mémoire et se déplacent vers le bas. Vous pouvez vous en rendre compte avec l'exemple suivant:

```
NEW
a=10.1:b=20:a$="Exemple 1":b$="Exemple 2"

PRINT@a,@b,@a$,@b$
```

Si vous examinez maintenant les adresses des variables, vous voyez que les variables numériques occupent 9 octets. Ici sont stockées les valeurs des variables a et b ainsi que leurs noms. Les variables alphanumériques (chaînes de caractères) sont cependant stockées différemment. Entrez par exemple:

```
PRINT PEEK(@a$)
```

Vous obtenez alors la valeur 9 qui correspond exactement à la longueur de la variable a$. Il en va de même pour la variable b$. Les valeurs des cases mémoire:

@a$+1 ainsi que:
@a$+2

constituent un pointeur. Ce pointeur indique au système d'exploitation où il peut trouver la chaîne de caractères elle-même. Cette méthode de stockage des chaînes vous semble peut-être compliquée mais elle présente un avantage décisif: il n'est pas nécessaire de déplacer la totalité de la table des variables lorsqu'une chaîne voit sa longueur modifiée; il suffit de modifier les trois octets que nous venons de décrire.

Mais faisons éditer la chaîne a$ de façon peu conventionnelle:

```
ad=PEEK(@a$+1)+256*PEEK(@a$+2)
```

Nous avons maintenant calculé l'adresse à laquelle est stockée la chaîne de caractères a$. Nous avons sauvé cette adresse dans la variable ad.

```
FOR I=0 TO PEEK(@a$)-1
PRINT CHR$(PEEK(ad+I));
NEXT I
```

Et vous obtenez exactement le contenu de la variable a$. Vous pouvez très facilement changer les valeurs des pointeurs des variables alphanumériques et arriver ainsi à ce que leur contenu se modifie à votre guise. Cette propriété sera aussi utilisée dans les programmes du chapitre 5 (routine d'interception des erreurs et gestion de fichier relatif). Détournez par exemple la chaîne a$ sur votre écran:

```
POKE@a$+1,&00
POKE@a$+2,&C1
```

Le pointeur de la chaîne de caractères est maintenant dirigé sur le beau milieu de la mémoire écran. Faites maintenant éditer la variable alphanumérique a$:

```
PRINT a$
```

Il n'est pas possible de décrire précisément l'effet obtenu car il sera différent suivant ce que contenait votre écran, mais il se peut que le cadre se mette à clignoter ou que le mode d'écriture change de lui-même. En tout cas vous obtenez une série de caractères indéfinis. Nous en terminerons ainsi avec cette introduction à l'instruction@.

1.5 SAUVEGARDE SEQUENTIELLE DE DONNEES

1.5.1 QU'EST-CE QUE LA SAUVEGARDE SEQUENTIELLE DE DONNEES?

Un lecteur de disquette ne doit bien sûr pas servir uniquement à la sauvegarde et au chargement des programmes; il convient en outre parfaitement à la sauvegarde de grandes masses de données qu'il était jusqu'alors, avec le lecteur de cassette, pratiquement impossible de maîtriser.

Le DDI-1 ne vous permet en principe de traiter que des fichiers SEQUENTIELS comme ceux que vous connaissez déjà d'après le Basic cassette; le principe est totalement identique. La sauvegarde séquentielle de données n'est pas la méthode la plus rapide mais c'est la plus simple pour sauvegarder des données. Elle est d'ailleurs tout à fait suffisante pour les petits ou moyens problèmes. Comme elle est d'autre part simple à comprendre, elle convient parfaitement à l'apprentissage.

Séquentiel ne signifie pas autre chose que CARACTERE par CARACTERE. Un fichier séquentiel est donc une séquence (une suite) de caractères (lettres, chiffres, caractères spéciaux etc.). Pour lire le dernier caractère d'un fichier, il faut lire tous les caractères situés auparavant, du premier à l'avant-dernier. Imaginez par exemple un livre de 100 pages. Pour lire la première lettre de la page 100 vous devriez d'abord lire TOUTES les lettres des pages 1 à 99. C'est ainsi qu'un fichier séquentiel est lu; une méthode évidemment peu pratique. Heureusement le DDI-1 est très rapide de sorte qu'on se rend à peine compte de tels problèmes.

Une possibilité nettement plus pratique pour la sauvegarde des données consiste à pouvoir accéder directement à un fichier, ce que l'on appelle accès direct ou Random access; le terme de fichier relatif s'est également imposé.

L'accès direct signifie, pour rester sur notre exemple du livre, que vous pouvez lire par exemple directement la 55ème lettre de la 29ème page sans être obligé de lire toutes les pages précédentes. Lorsque vous en êtes arrivé, sur un fichier séquentiel, au 100ème enregistrement, vous ne pouvez plus lire de caractère appartenant au 5ème enregistrement, il n'est donc pas possible de revenir en arrière. C'est là une des limites à l'efficacité des fichiers séquentiels que l'on ne rencontre plus avec les

fichiers relatifs.

Il s'agit cependant simplement ici pour nous de vous faire comprendre grossièrement les différences entre les deux types de fichier. Vous trouverez au chapitre 5 de plus amples informations sur les fichiers relatifs et sur la façon dont ils peuvent être programmés sur le DDI-1.

Nous allons maintenant expliquer avec quelques exemples comment on peut utiliser de façon efficace la sauvegarde séquentielle des données puisque celle-ci constitue (pour le moment) la seule possibilité dont nous disposons pour sauvegarder des données sur disquette.

Nous allons maintenant créer notre premier petit fichier. Nous allons, à titre d'exemple, créer un agenda téléphonique qui nous permettra d'enregistrer les principaux numéros de téléphone de nos parents ou amis.

Lorsqu'on crée un fichier, se pose toujours au début la question de savoir ce qu'on veut au juste enregistrer. Dans l'exemple suivant nous avons choisi les indications suivantes:

1)Nom (=Champ 1)
2)Prénom (=Champ 2)
3)No de tél. (=Champ 3)

Un enregistrement est une unité cohérente. Chaque enregistrement contient des informations que l'on retrouve également dans tout autre enregistrement. Notre enregistrement se compose de trois champs.

Nous pourrions ainsi avoir un enregistrement "Dupont" qui contienne comme informations les nom, prénom et numéro de téléphone de notre collègue Dupont.

Figure 3

LE FICHIER COMPLET

Enregistrement 1	Enregistrement 2	Enregistrement 3
Dupont	Jourdain	Peterson
Thomas	Cécile	Hans-Dieter
23 44 98 21	80 23 44 01	34 41 22 37

Nous avons trois champs par enregistrement:

Nom, Prénom et Numéro de téléphone

Champ 1: Nom
Champ 2: Prénom
Champ 3: Numéro de téléphone

Vous pouvez bien vous représenter, au vu de cette figure, les subdivisions que comporte un fichier.
Lorsque les données sont écrites dans un fichier, il n'est pas possible de distinguer aussi nettement que nous l'avons fait ici sur le papier les subdivisions en enregistrements. Seuls les champs sont séparés physiquement les uns des autres. La séparation entre les enregistrements ne se distingue pas en effet de la séparation entre les champs de données; c'est le même symbole de séparation qui est utilisé. La séparation entre les différents enregistrements, c'est-à-dire la bonne affectation des champs de données incombe au programmeur. Il est le seul à savoir combien de champs comporte chaque enregistrement.

Lorsque vous avez terminé une entrée à l'écran, vous actionnez la touche ENTER, également appelée RETOUR DE CHARIOT, en anglais CARRIAGE RETURN. Il en va exactement de même avec la sauvegarde séquentielle des données puisque les différents champs de données sont en effet séparés sur la disquette par un retour de chariot: le code ASCII 13 est produit.

Pour vous permettre de vous représenter plus facilement comment les différents champs de données sont séparés, examinons la figure 4. Le symbole de séparation retour de chariot est représenté par l'étoile (*).

Figure 4

Dupont*Thomas*23449821*Jourdain*C...

Champ1 Champ2 Champ 3 Champ 1 ...

ENREGISTREMENT 1 ENREGISTREMENT 2

Vous voyez d'après notre exemple que les champs de données ne doivent pas avoir obligatoirement la même longueur mais qu'ils peuvent avoir dans les fichiers séquentiels une longueur quelconque, il en va de même pour la longueur des enregistrements. Les champs de données peuvent être cependant clairement identifiés puisqu'ils sont séparés les uns des autres par le symbole de retour de chariot (*).

Pour lire un tel fichier, on utilise l'instruction INPUT#9.

```
OPENIN"Nom de fichier"
INPUT#9,Nom$,Prenom$,Telephone$
CLOSEIN
```

Après ces instructions, seul un enregistrement déterminé serait lu; les variables pourraient avoir par exemple les valeurs suivantes:

```
Nom$         ="Durand"
Prenom$      ="Alain"
Telephone$   ="14 22 51 33"
```

Les guillemets ne servent ici qu'à indiquer le début et la fin d'une chaîne de caractères et ils n'appartiennent pas à la chaîne elle-même. Des chaînes peuvent toutefois contenir des guillemets.

1.5.2 LA PROGRAMMATION DES FICHIERS SEQUENTIELS

Mais nous allons maintenant créer concrètement un fichier séquentiel sur disquette. Entrez pour cela le petit programme suivant:

```
10 REM ================================
20 REM    PREMIER PROGRAMME DE FICHIER
30 REM ================================
40 :
50 OPENOUT"Test.dat"
60 PRINT#9,"Durand"
70 PRINT#9,"Louis"
80 PRINT#9,"14225114"
90 REM --- premier enregistrement ---
100 PRINT#9,"Chalet"
110 PRINT#9,"Lucienne"
120 PRINT#9,"23990123"
130 REM --- second enregistrement ---
140 PRINT#9,"Lefol"
150 PRINT#9,"Charles"
160 PRINT#9,"18155118"
170 REM --- troisieme enregistrement ---
180 CLOSEOUT
190 END
```

Nous avons créé un fichier sous le nom de "Test.dat" qui contient trois enregistrements.
L'instruction Basic PRINT#n s'emploie exactement comme l'instruction PRINT normale. Vous connaissez certainement déjà l'instruction PRINT# en liaison avec les windows (fenêtres) que vous pouvez définir sur votre Amstrad. L'instruction PRINT# vous permet d'appeler une des instructions définies lorsque n est compris entre 0 et 7. Si n=8, c'est l'imprimante qui est appelée et si n=9 c'est le lecteur de cassette ou le lecteur de disquette, suivant la configuration utilisée. Il en va de même pour les instructions INPUT#n et LINE INPUT#n.

Le caractère de séparation logique Carriage Return est dans notre exemple placé entre les enregistrements parce que l'instruction PRINT#9 n'est pas suivie d'un point-virgule. C'est donc le même principe que pour l'instruction PRINT simple. Si vous placez un point-virgule à la fin d'une instruction PRINT, la prochaine sortie sera écrite sur la même ligne. Pour les sorties sur le lecteur de disquette, vous pouvez également constituer un champ de données en plaçant plusieurs sorties à la suite l'une de l'autre.

Mais commençons d'abord par relire nos données:

```
10 REM ================================
20 REM      LIRE LES DONNEES SAUVEES
30 REM ================================
40 :
50 OPENIN"Test.dat"
60 INPUT#9,Nom$,Prenom$,Telephone$
70 PRINT Nom$,Prenom$,Telephone$
80 INPUT#9,Nom$,Prenom$,Telephone$
90 PRINT Nom$,Prenom$,Telephone$
100 INPUT#9,Nom$,Prenom$,Telephone$
110 PRINT Nom$,Prenom$,Telephone$
120 CLOSEIN
130 END
```

Vous obtenez alors la sortie suivante:

```
Durand        Louis     14225114
Chalet        Lucienne  23990123
Lefol         Charles   18155118
```

Nous avons donc réussi à sauvegarder des données sur disquette et à les relire; nous pouvons ainsi dire pour récapituler:

Avec PRINT#9 nous écrivons des données sur disquette et avec INPUT#9 nous pouvons les relire. L'entrée et la sortie sur disquette ne se distinguent donc pas trop de l'entrée et de la sortie sur écran.

Notre exemple fonctionne donc bien mais de façon encore très peu pratique si l'on pense au travail qui serait nécessaire pour faire lire ainsi 100 enregistrements. C'est une boucle qui sera ici la solution de notre problème et qui apportera un allègement considérable:

```
10 REM ==================================
20 REM       LIRE LES DONNEES (2)
30 REM ==================================
40 :
50 OPENIN"Test.dat"
60 FOR i=1 TO 3
70 INPUT#9,Nom$,Prenom$,Telephone$
80 PRINT Nom$,Prenom$,Telephone$
90 NEXT i
100 CLOSEIN
110 END
```

Il n'est pas nécessaire de toujours lire un enregistrement d'une seule traite. Vous pourriez également procéder ainsi:

```
10 REM ==================================
20 REM       LIRE LES DONNEES (3)
30 REM ==================================
40 :
50 OPENIN"Test.dat"
60 FOR i=1 TO 3
70 INPUT#9,Nom$
80 PRINT Nom$,
90 INPUT#9,Prenom$
100 PRINT Prenom$,
110 INPUT#9,Telephone$
120 PRINT Telephone$
130 NEXT i
140 CLOSEIN
150 END
```

Vous devez vous dire qu'après l'ouverture d'un fichier, les données du fichier sont à votre disposition. Peu importe à cet égard que vous lisiez un enregistrement avec une seule instruction INPUT, comme par exemple avec:

INPUT#9,Nom$,Prenom$,Telephone$

ou que vous lisiez l'enregistrement morceau par morceau avec plusieurs instructions Input, comme par exemple avec:

INPUT#9,Nom$
INPUT#9,Prenom$

```
INPUT#9,Telephone$
```

Lorsque vous venez de lire un enregistrement, un pointeur note l'emplacement où doit se poursuivre la lecture.

Vous pouvez également fermer prématurément un fichier que vous avez ouvert en lecture, sans en avoir donc lu tous les enregistrements.

Si vous avez cependant déjà lu le dernier élément du fichier et que vous effectuez à nouveau une nouvelle tentative de lecture, vous obtenez le message d'erreur:

EOF met

Nous allons vous montrer cela dans un exemple.

Chaque chiffre représente, dans l'exemple suivant, un enregistrement:

```
12345678901234567890123*
```

Nous avons donc 23 enregistrements; EOF est représenté par le caractère "*". Après l'ouverture du fichier en lecture, le premier enregistrement sera lu. Le pointeur interne est simulé dans la seconde ligne. Voici comment les choses se présentent de façon interne après l'instruction OPENIN:

```
12345678901234567890123*
↑
```

Le pointeur est dirigé sur le premier enregistrement; nous lisons maintenant le premier enregistrement avec l'instruction INPUT. Après que le premier enregistrement ait été lu, notre pointeur interne est dirigé sur le second enregistrement:

```
12345678901234567890123*
 ↑
```

Lors d'une lecture, c'est maintenant le second enregistrement qui serait lu. Cela continue ainsi jusqu'à ce que nous en arrivions au dernier enregistrement de notre fichier.

```
1234567890123456789O123*
                       ↑
```

Dans notre exemple, le dernier enregistrement n'a donc pas été encore lu puisque le pointeur interne est dirigé sur ce dernier enregistrement. Nous lisons maintenant également ce dernier enregistrement:

```
1234567890123456789O123*
                        ↑
```

Il n'y a plus maintenant d'enregistrement disponible, nous sommes arrivé à la fin des entrées du fichier. Le pointeur est maintenant dirigé sur l'étoile qui est censée représenter ici la marque EOF. Lors de notre dernière tentative de lecture, le flag EOF a été mis par le système d'exploitation sur vrai=-1. Une autre tentative de lecture provoquerait l'erreur EOF met.

La fonction EOF nous indique donc si nous avons ou non déjà lu le dernier élément d'un fichier.

Pour éviter de provoquer cette erreur, nous pouvons utiliser la fonction Basic EOF. Quand vous ne savez pas exactement combien d'enregistrements vous devez lire, il est même indispensable d'utiliser la fonction EOF. Notre programme se présenterait alors ainsi:

```
10 REM ================================
20 REM  LIRE LES DONNEES AVEC EOF
30 REM ================================
40 :
50 OPENIN"Test.dat"
60 WHILE NOT EOF
70 INPUT#9,Nom$,Prenom$,Telephone$
80 PRINT Nom$,Prenom$,Telephone$
90 WEND
100 CLOSEIN
110 END
```

Cet exemple est cependant assez dangereux car nous partons du principe qu'il reste toujours encore trois autre champs de données tant que nous n'avons pas rencontré EOF. Pour les fichiers de structure inconnue, il

est impossible de le prévoir. Notre programme devrait donc se présenter ainsi:

```
10 REM ================================
20 REM  LECTURE DE DONNEES SOUPLE
30 REM ================================
40 :
50 OPENIN"Test.dat"
60 WHILE NOT EOF
70 INPUT#9,Champ$
80 PRINT Champ$
90 WEND
100 CLOSEIN
110 END
```

Avec ce programme, une erreur EOF met est exclue puisqu'on teste si EOF après la lecture de chaque champ.

Quand les champs de données sont lus avec l'instruction INPUT#9, les caractères suivants fonctionnent comme symboles de séparation entre les champs de données:

Carriage Return (retour de chariot)
Virgule (,)
Marque EOF

Lorsque vous lisez des variables numériques avec INPUT#9, la <barre espace> fait également office de marque de séparation.

Ces marques de séparation sont également faciles à retenir puisque la virgule et le retour de chariot font également office de marques de séparation avec l'instruction INPUT normale.

Bien sûr vous pourriez également définir une marque de fin "particulière", vous pourriez par exemple écrire toujours dans le dernier champ de donnée un "*fin" ou autre. Vous pourriez alors faire rechercher cette marque par le programme. Vous m'accorderez cependant certainement que la manipulation de la fonction EOF constitue une méthode plus sûre et plus simple.

Le plus souvent, on ne veut pas simplement lire des données pour les sortir immédiatement mais les données doivent encore pouvoir rester dans

la mémoire de l'ordinateur pour pouvoir être évaluées et modifiées. Dans ce cas, le mieux est de constituer un TABLEAU (ARRAY en anglais), une variable indexée. Si cette notion vous est inconnue, nous vous invitons à vous reporter au chapitre correspondant du manuel du Basic.
Notre programme d'exemple se présenterait maintenant comme suit:

```
10 DIM Nom$(3),Prenom$(3),Telephone$(3)
20 REM ================================
30 REM  LIRE-SAUVEGARDER FICHIER
40 REM ================================
50 :
60 x=0
70 OPENIN"Test.dat"
80 WHILE NOT EOF
90 x=x+1
100 INPUT#9,Nom$(x),Prenom$(x),Telephone$(x)
110 WEND
120 PRINT x-1;"enregistrements ont ete lus."
130 CLOSEIN
140 END
```

Dans l'instruction DIM en ligne 10, vous devez définir la limite de l'index en fonction de la limite supérieure du nombre d'enregistrements.

Avec une telle boucle, on peut lire un fichier et le placer dans un tableau. On peut ensuite traiter le fichier par le programme puis sauvegarder à nouveau le fichier de façon séquentielle. C'est ici le principe LECTURE/ENTREE, TRAITEMENT et SORTIE qui s'applique. Quand la mémoire le permet, nous vous recommandons de faire lire vos fichiers de cette façon et de les faire traiter dans le programme. Cette méthode a de grands avantages en ce qui concerne la vitesse de traitement.

1.5.3 LES FICHIERS SEQUENTIELS ET LES TABLEAUX

Sur les programmes d'envergure qui doivent gérer de grandes masses de données, il est conseillé de déterminer auparavant précisément la masse de données prévue et de définir alors cette masse de données dans le programme.

Si vous lisez le fichier séquentiel et que vous placiez tous les enregistrements dans un tableau, vous économisez beaucoup de temps de traitement car à partir de ce moment vous n'avez plus à accéder au

lecteur de disquette. (Un accès à des tableaux est toujours plus rapide qu'un accès à la disquette).

Nos tableaux se présentent ainsi:

No d'index	Nom$	Prenom$	Telephone$
1	Durand	Louis	14225114
2	Chalet	Lucienne	23990123
3	Lefol	Charles	18155118

Nous avons donc simplement lu une fois ces données et nous les avons placées dans des tableaux. Nous avons ici utilisé 3 différents tableaux à une dimension. Il est plus pratique pour le programmeur de créer un tableau à deux dimensions de façon à pouvoir lui donner un SEUL nom, ce qui augmente la clarté du programme.

	Champ 1	Champ 2	Champ 3
Enregistrement 1	D$(1,1)	D$(1,2)	D$(1,3)
Enregistrement 2	D$(2,1)	D$(2,2)	D$(2,3)
Enregistrement 3	D$(3,1)	D$(3,2)	D$(3,3)
Enregistrement 4	D$(4,1)	D$(4,2)	D$(4,3)
Enregistrement 5	D$(5,1)	D$(5,2)	D$(5,3)

Notre tableau D$(x,y) a 5 enregistrements de 3 champs chacun. Ce tableau doit être "dimensionné", c'est-à-dire créé avec l'instruction:

DIM D$(5,3)

Le tableau D$(x,y) pourrait avoir par exemple le contenu suivant:

D$(x,1)=Nom	auparavant Nom$(x)
D$(x,2)=Prénom	auparavant Prénom$(x)
D$(x,3)=Téléphone	auparavant Téléphone$(x)

Notre programme d'exemple se présente alors ainsi:

```
10 REM ================================
20 REM  LIRE LES DONNEES AVEC D$(x,y)
30 REM ================================
40 DIM d$(3,3)
50 REM 3 enregistrements et 3 champs
60 OPENIN"Test.dat"
80 FOR i=1 TO 3
90  FOR j=1 TO 3
100  INPUT #9,d$(i,j)
110  NEXT j
120 NEXT i
130 :
140 REM ==============================
150 REM              SORTIE
160 REM ==============================
170 FOR i=1 TO 3
180  FOR j=1 TO 3
190  PRINT d$(i,j)
200  NEXT j
210  PRINT
220 NEXT i
230 CLOSEIN
240 END
```

Vous pouvez prendre ce programme d'exemple comme modèle si vous voulez vous-même écrire un programme de fichier. En imbriquant deux boucles vous réalisez une lecture logique pas à pas des champs de données et des enregistrements. Lorsque vous voulez modifier des données dans un programme de fichier, le mieux est de lire toutes les données et de les placer dans un tel tableau. Vous pouvez alors traiter les données, les corriger, etc. A la fin du programme, les données doivent alors être à nouveau sauvegardées sous le même nom.
Le plus souvent on crée également un fichier auxiliaire dans lequel se trouvent des informations sur la masse de données. Par masse de données vous devez comprendre: combien de données comporte le tableau? Combien de champs de données comporte chaque enregistrement? Lorsque vous créez un programme souple de fichier ceci est indispensable pour des raisons pratiques. D'autre part vous pouvez de cette façon économiser une précieuse place mémoire car toute coordonnée inutile coûte de la place mémoire et du temps. L'instruction DIM permet également un

dimensionnement dynamique, c'est-à-dire que vous pouvez utiliser des variables comme limites d'index. Nous allons maintenant créer un tel fichier d'information. Nous entrerons à cet effet le nombre d'ENREGISTREMENTS ainsi que le nombre de CHAMPS de données par enregistrement.

```
OPENOUT"Test.inf"
WRITE#9,3,3
CLOSEOUT
```

Nous appellerons le type de fichier "INF" pour fichier d'INFormation. Le nom de fichier doit être dans notre exemple identique au nom du fichier (.dat). Voici cette routine souple de lecture:

```
10 REM ================================
20 REM  ROUTINE SOUPLE DE LECTURE
30 REM ================================
40 :
50 INPUT"Nom du fichier: ";file$
60 OPENIN file$+".inf"
70  INPUT #9,enregistrements,champs
80 CLOSEIN
90 :
100 DIM d$(enregistrements,champs)
110 :
120 OPENIN file$+".dat"
130 FOR i=1 TO enregistrements
140  FOR j=1 TO champs
150  INPUT#9,d$(i,j)
160  NEXT j
170 NEXT i
180 CLOSEIN
190 END
```

Lancez le programme et entrez "Test".

En ligne 100 le dimensionnement s'effectue alors en fonction des besoins. Dans les cas où le nombre d'enregistrements peut être augmenté au cours du déroulement du programme, il faut avoir prévu cette éventualité lors du dimensionnement. Une autre solution consiste à fixer auparavant le nombre maximal d'enregistrements; le nombre de champs peut cependant rester malgré tout dynamique car il est peu probable que le nombre des

champs de données ait à être modifié au cours de l'exécution du programme. La lecture à partir d'un fichier des nombres d'enregistrements et de champs peut être cependant malgré tout d'une aide précieuse. Si vous remplacez la ligne 100 par:

```
100 DIM d$(200,champs)
```

vous obtenez bien un nombre de champs dépendant de chaque fichier.

Nous avons jusqu'à présent toujours indiqué, lors de l'ouverture d'un fichier, un nom précis. En effet le nom de fichier n'était pas variable, il figurait entre guillemets et il était déjà fixé lors de la programmation. On ouvre toutefois parfois un fichier avec une variable alphanumérique comme nous l'avons fait dans notre exemple (lignes 60 et 120). Malheureusement le système d'exploitation n'est pas entièrement au point à cet égard. AMSDOS ouvre en effet parfois dans ce cas les fichiers sous un nom incorrect ou réagit par un message d'erreur. Si vous faites alors afficher, après ce message d'erreur, le contenu de la variable alphanumérique correspondante, vous pouvez vous rendre compte que le contenu de cette variable est pourtant correct.
Lors de l'ouverture d'un fichier un buffer de 4096 octets est créé dans la RAM de l'Amstrad CPC. Ce buffer est situé la plupart du temps dans les régions supérieures de la mémoire. C'est cependant également dans ces régions que se trouvent les variables alphanumériques. Il peut donc arriver assez facilement que des "collisions" se produisent ici, surtout lorsque vous travaillez avec des tableaux car le nombre de chaînes de caractères est dans ce cas particulièrement élevé. Cela tient au fait que le système d'exploitation "nettoie" de temps à autre la mémoire car les manipulations de chaînes produisent beaucoup de déchet. Nous entendons par déchet tous anciens morceaux de chaîne qui sont devenus inutiles à la suite de manipulations de chaînes. Ce "nettoyage" est appelé GARBAGE COLLECTION, il s'agit d'une sorte de réorganisation de la mémoire qui réorganise la mémoire de chaînes de caractères. Il peut se produire dans ce cas que le buffer disquette soit victime de cette réorganisation. Lorsque le buffer est décalé par la Garbage Collection, le nom de fichier ne peut plus être lu correctement. La petite routine suivante vous permet de remédier à cet inconvénient:

```
OPENOUT"Dummy"
MEMORY HIMEM-1
CLOSEOUT
```

Il est conseillé de placer cette instruction au début de chaque programme qui travaille avec la disquette. Vous devez utiliser cette instruction avant de définir la première variable alphanumérique. Le buffer de 4096 octets est créé D'AVANCE et cette zone est protégée avec l'instruction MEMORY. A partir de ce moment, le buffer disquette ne pourra plus être détruit par la Garbage Collection car il sera protégé par l'instruction MEMORY.

Le nom "Dummy" dans l'instruction OPENOUT ne désigne pas dans ce cas un fichier (puisque rien n'y sera écrit) mais il correspond dans le langage technique informatique à une sorte de "poubelle électronique". L'instruction OPENOUT"Dummy" a en fait pour effet de créer le buffer de 4096 octets dont nous avons parlé. La variable système HIMEM est alors diminuée de façon à ce que le buffer soit protégé des variables. Après un CLOSEOUT, HIMEM serait à nouveau relevé, mais pour éviter cela l'instruction MEMORY a été employée.

Si donc vous avez obtenu un message d'erreur en faisant exécuter notre programme d'exemple qui ne contenait pas encore cette routine, ajoutez la ligne 40 suivante:

```
40 OPENOUT"Dummy":MEMORY HIMEM-1:CLOSEOUT
```

Le risque existe maintenant encore que l'utilisateur entre un nom de fichier incorrect ce qui entraînerait l'interruption du programme. Mais cela peut également être évité, par exemple en créant un fichier dans lequel seront enregistrés tous les noms de fichier; ce serait en fait un "catalogue privé". Nous appellerons ce fichier "Filename.dir".

```
OPENOUT"Filename.dir"
PRINT#9,"Test"
CLOSEOUT
```

Notre catalogue privé ne contient pour le moment que l'entrée "Test". Nous n'avons plus maintenant qu'à intégrer encore cette routine de test et un message d'erreur sera exclu.

```
10 REM ================================
20 REM  ROUTINE SOUPLE DE LECTURE
30 REM ================================
40 OPENOUT"Dummy":MEMORY HIMEM-1:CLOSEOUT
50 INPUT"Nom du fichier: ";file$
60 GOSUB 1000
70 IF trouve=0 THEN 50
80 OPENIN file$+".inf"
90  INPUT #9,enregistrements,champs
100 CLOSEIN
110 :
120 DIM d$(200,champs)
130 :
140 OPENIN file$+".dat"
150 FOR i=1 TO enregistrements
160  FOR j=1 TO champs
170  INPUT#9,d$(i,j)
180  NEXT j
190 NEXT i
200 CLOSEIN
210 END
1000 REM ==============================
1010 REM   Tester si nom autorise
1020 REM ==============================
1030 OPENIN"Filename.dir"
1040 trouve=0
1050 WHILE NOT EOF AND trouve=0
1060  INPUT#9,nom$
1070  IF nom$=file$ THEN trouve=1
1080 WEND
1090 CLOSEIN
1100 RETURN
```

Un tel test ou un test semblable est indispensable pour qu'un programme soit d'un usage pratique. Les entrées erronnées devraient, autant que possible, toujours être identifiées et écartées.

Il n'est possible d'ouvrir qu'un SEUL canal d'entrée et un SEUL canal de sortie. Vous pouvez donc ouvrir au maximum deux canaux en direction du lecteur de disquette.

Dans la sous-routine commençant en ligne 1030, on teste s'il existe un fichier portant le nom file$. A cet effet, toutes les entrées du fichier "Filename.dir" sont comparées à la variable file$. Si la routine rencontre la fin du fichier sans avoir trouvé le nom recherché, la variable "trouve" vaut 0. En cas de succès, cette variable est mise sur 1. Cette valeur peut alors être testée dans le programme principal qui réagira comme il convient.

Si vous voulez par exemple écrire vous même un programme de gestion d'adresses pratique, nous ne pouvons que vous conseiller d'intégrer de telles routines de protection.

Cela suppose bien sûr que le fichier "Filename.dir" existe; si ce n'est pas le cas, un message d'erreur "File not found" sera sorti. Pour éviter cela, on pourrait prévoir un point du menu qui pourrait s'appeler "Initialisation" et qui créerait ce fichier.

Toutes ces "astuces" sont contenues dans un petit programme de gestion de fichier que vous trouverez au chapitre 5 de cet ouvrage. Nous vous conseillons d'étudier les passages correspondants de ce programme de façon à ce que vous compreniez bien la manipulation de ces routines.

1.5.4 DIFFERENCES ENTRE PRINT# ET WRITE#

Nous avons jusqu'ici utilisé uniquement l'instruction PRINT#. Vous avez cependant certainement déjà vu, aussi bien dans le manuel d'utilisation que dans le présent ouvrage, qu'il existe aussi une instruction WRITE#.

Dans nos programmes d'exemple c'est le Carriage Return qui était utilisé comme marque de séparation. Vous vous souvenez cependant certainement que le Carriage Return n'est pas le seul symbole de séparation et que la virgule (,) fait également office de symbole de séparation, aussi bien pour les chaînes de caractères que pour les données numériques. Pour les

données numériques il est en fait sans importance que la séparation soit faite avec des virgules, des Carriage Return ou même des espaces. Par contre, avec les chaînes de caractères, cela peut avoir un certain nombre d'effets désagréables. Essayez l'exemple suivant:

```
10 REM ================================
20 REM      EXEMPLE POUR PRINT #
30 REM ================================
40 :
50 OPENOUT"Demo"
60 PRINT#9,"Chalet,Lucienne"
70 PRINT#9,"Lefol,Charles"
80 CLOSEOUT
90 :
100 REM ================================
110 REM      ET LIRE A NOUVEAU
120 REM ================================
130 :
140 OPENIN"Demo"
150 INPUT#9,Nom1$
160 INPUT#9,Nom2$
170 CLOSEIN
180 PRINT"Nom1 = "Nom1$
190 PRINT"Nom2 = "Nom2$
200 END
```

Vous vous attendez certainement à ce que la sortie sur l'écran se présente ainsi:

Nom1=Chalet, Lucienne
Nom2=Lefol, Charles

Certains d'entre vous le savent certainement déjà: le résultat produit ne se présente malheureusement pas ainsi. Si vous lancez le programme, vous pouvez constater ce phénomène mystérieux:

Nom1=Chalet
Nom2=Lucienne

Vous voyez très nettement d'après cet exemple que la virgule fait office de symbole de séparation et que cela a entraîné avec l'instruction INPUT#9 une séparation entre les deux chaînes de caractères. D'autre

part, il manque la virgule de la chaîne de caractères ainsi que l'espace devant "Lucienne". La solution de ce problème est très simple: quand on utilise l'instruction WRITE#, une chaîne à sortir est placée entre guillemets. Lors de la lecture de cette chaîne tous les espaces, même placés au début, ainsi que toutes les virgules seront acceptés dans la chaîne. Seuls les guillemets ne seront pas et n'ont pas à être acceptés dans la chaîne.

Vous pouvez maintenant faire sortir sur l'écran différentes valeurs pour mieux comprendre le mode de fonctionnement de cette instruction. Entrez par exemple:

```
WRITE 1,2,"Oui,oui,c'est comme ca.",3
```

Sur l'écran apparaît:

```
1,2,"Oui,oui,c'est comme ca.",3
```

Si vous faites sortir le même texte avec l'instruction PRINT, l'image suivante apparaît sur le moniteur:

```
1,2 Oui,oui,c'est comme ca. 3
```

En effet, comme vous le savez, la virgule a pour effet avec l'instruction PRINT de faire sauter le tabulateur à la prochaine tabulation, de sorte que des espaces vides importants apparaissent. L'écriture sur disquette s'effectue exactement comme sur l'écran, c'est-à-dire que les espaces vides correspondants sont également écrits sur la disquette (ce qui d'ailleurs constitue un gaspillage de la place mémoire disponible).

On peut cependant aisément constater que la chaîne n'est plus traitée comme une chaîne unique, à cause des virgules. Avec l'instruction WRITE, les guillemets marquent le début et la fin de la chaîne de façon parfaitement claire.

Un effet secondaire appréciable est l'économie de place mémoire sur la disquette lorsqu'on veut sauvegarder des variables numériques.

On pourrait également reconstituer l'instruction WRITE avec l'instruction PRINT:

```
PRINT 1",";2;",",chr$(34);"Oui,oui,c'est comme ca.";chr$(34);",";3
```

Le résultat sur l'écran ou sur la disquette sera identique avec cette façon d'écrire et l'on obtiendra également en lecture l'effet recherché. Mais l'instruction WRITE est plus simple et plus pratique et c'est pourquoi il est recommandé d'y avoir recours le plus souvent possible. Elle est particulièrement pratique lorsqu'on veut écrire plusieurs champs de données en une ligne Basic sur la disquette. Comme vous pouvez le constater d'après le programme d'exemple, nous avons stocké sur la disquette un seul champ de données par ligne PRINT# et ainsi, comme aucun point-virgule ne figure à la fin de l'instruction PRINT#, nous avons fait envoyer un Carriage Return comme symbole de séparation. Avec l'instruction WRITE, notre programme devient donc plus court et plus clair. Il se présente maintenant ainsi:

```
10 REM =================================
20 REM      EXEMPLE POUR WRITE #
30 REM =================================
40 :
50 OPENOUT"Demo"
60 WRITE#9,"Chalet,Lucienne"
70 WRITE#9,"Lefol,Charles"
80 CLOSEOUT
90 :
100 REM ================================
110 REM     ET LIRE A NOUVEAU
120 REM ================================
130 :
140 OPENIN"Demo"
150 INPUT#9,Nom1$
160 INPUT#9,Nom2$
170 CLOSEIN
180 PRINT"Nom1 = "Nom1$
190 PRINT"Nom2 = "Nom2$
200 END
```

Après avoir lancé le programme, vous constatez que nous sommes maintenant arrivé au résultat voulu.

Après une instruction PRINT# ou WRITE# une marque de séparation est donc AUTOMATIQUEMENT envoyée qui est le Carriage Return. Il y a cependant des cas où cet automatisme peut être gênant, par exemple lorsque vous voulez effectuer des opérations sur les chaînes de caractères pour ne les sortir que plus tard. Nous allons maintenant sortir tout l'alphabet en ne

résolvant pas cependant ce problème avec une simple instruction PRINT# mais en programmant une boucle.

```
10 REM ================================
20 REM     EXEMPLE DE CONCATENATION
30 REM ================================
40 :
50 OPENOUT"Demo"
60 FOR i=1 TO 26
70 PRINT#9,CHR$(64+i);
80 NEXT i
90 PRINT#9
100 CLOSEOUT
110 OPENIN"Demo"
120 INPUT#9,a$
130 PRINT a$
140 CLOSEIN
150 END
```

Nous définissons en ligne 60 une variable de boucle i, qui comptera de 1 à 26. L'alphabet a en effet 26 lettres. Le code ASCII de "A" est 65, de sorte que nous pouvons calculer en ligne 70 le code ASCII du caractère à sortir en ajoutant la valeur actuelle de la variable de boucle au décalage 64. L'important est ici que le dernier caractère en ligne 70 soit un point-virgule. Exactement comme en Basic lorsqu'on effectue une sortie sur écran, le point-virgule a pour effet sur la disquette qu'aucune "fin de champ" n'est envoyée, de même qu'à l'écran il manquerait la "fin de ligne".

Après avoir lancé le programme, nous voyons que la variable a$ contient tout l'alphabet. Cela met bien en évidence la signification du point-virgule. Dans cet exemple, il ne faut en aucun cas utiliser l'instruction WRITE car la sortie sur écran se présenterait alors ainsi:

"A""B""C""D""E""F""G""H"...."Y""Z"*

alors que nous obtenons dans notre exemple le résultat suivant:

ABCDEFGH....YZ*

(* réprésente à nouveau le Carriage Return)

Vous voyez qu'on doit se demander pour chaque application laquelle des deux instructions convient le mieux. Il est souvent indifférent que vous utilisiez PRINT#9 ou WRITE#9. WRITE#9 est parfois plus intéressant, mais parfois aussi cette instruction est totalement inadaptée, comme dans l'exemple que nous venons de donner.

Essayez de deviner quelle valeur la variable a$ aurait si on remplaçait la ligne 70 par:

```
70 WRITE#9,CHR$(i+64);
```

Lorsque vous penserez avoir trouvé la solution, modifiez la ligne et lancez le programme. Aviez-vous prévu juste? Vous pouvez maintenant passer au chapitre suivant.

1.5.5 DIFFERENCES ENTRE INPUT# ET LINE INPUT#

Vous connaissez maintenant les différences qu'il y a entre PRINT#9 et WRITE#9. Pour la lecture des données, il existe également deux instructions différentes: l'instruction normale INPUT#9 que nous avons toujours utilisée jusqu'ici, et l'instruction LINE INPUT#9. Ici aussi il y a des différences essentielles.

Nous nous limiterons, dans notre comparaison entre INPUT et LINE INPUT, aux variables alphanumériques. Les symboles de séparation peuvent être: le retour de chariot, la virgule et la marque EOF. Nous ne pouvons donc lire, avec INPUT#9, aucune chaîne contenant une virgule (sauf bien sûr si cette chaîne avait été placée entre guillemets) car cette virgule sera automatiquement interprétée comme une marque de séparation. Il en résulte un autre problème: lorsque les guillemets marquent une chaîne, comment puis-je alors lire ces guillemets? Il est impossible de lire les guillemets dans une chaîne avec INPUT, cela nous le savons.

C'est ici qu'intervient l'instruction LINE INPUT. LINE INPUT lit tous les caractères; seul le retour de chariot fait office de marque de séparation, les virgules et les guillemets sont donc également acceptés par LINE INPUT.

Lorsque vous utilisez l'instruction:

WRITE#9,1,2,"Oui,oui, c'est comme ca.",3

pour écrire sur la disquette et que vous relisez cela avec:

LINE INPUT#9,tout$

la variable tout$ contiendra:

1,2,"Oui,oui, c'est comme ca.",3

TOUS les symboles de séparation sans exception ont donc été acceptés dans la chaîne, les guillemets y compris.

LINE INPUT n'est pas toujours supérieur à l'instruction INPUT; cela pourrait même le plus souvent avoir des conséquences catastrophiques si vous remplaciez simplement, dans un programme existant, INPUT#9 par LINE INPUT#9. Vous vous rendez certainement compte que cela est lié au fait que les symboles de séparation sont alors ignorés. Si l'on ne tient pas compte dans l'instruction PRINT ou l'instruction WRITE du fait que la lecture se fera avec LINE INPUT, et donc si l'on n'utilise pas UNIQUEMENT le retour de chariot comme marque de séparation, on peut aboutir à des résultats assez surprenants. Lorsque vous écrivez vous-même un programme qui accède souvent à la disquette, réfléchissez bien avant de choisir quelles instructions vous utiliserez en écriture et en lecture.

Mais voici maintenant encore un exemple où l'instruction LINE INPUT est indispensable parce qu'il s'agit de lire un fichier dont le contenu et la structure sont inconnus.

Chargez un des programmes réalisés dans le présent ouvrage et sauvegardez-le immédiatement avec:

```
OPENOUT"ASCII.DAT"
LIST#9
CLOSEOUT
```

Vous avez maintenant sauvegardé le listing du programme sur la disquette, sous le nom de "ASCII.DAT". Un fichier de programme normal ne peut pas être ouvert en lecture avec l'instruction OPENIN car un tel fichier a un header (=une tête de fichier) qui n'est pas accepté par l'instruction

OPENIN. Après que vous ayez donc LISTé un fichier de programme sur la disquette, entrez le petit programme suivant:

NEW

```
10 REM ================================
20 REM LECTURE D'UN FICHIER QUELCONQUE
30 REM ================================
40 :
50 OPENIN"ASCII.dat"
60 WHILE NOT EOF
70  LINE INPUT#9,ligne$
80  PRINT ligne$
90 WEND
100 CLOSEIN
110 END
```

Cette routine affiche le listing de votre programme sur l'écran avec tous les guillemets et toutes les virgules, comme dans le programme original. Pour des tâches semblables, l'instruction LINE INPUT est parfaitement appropriée puisqu'elle va chercher sans problème tous les caractères sur la disquette.

Malheureusement le Basic de l'Amstrad ne permet pas de lire des caractères isolés dans un fichier ouvert en lecture, comme on peut le faire dans d'autres version du Basic, sur d'autres ordinateurs. Cette possibilité aurait bien sûr l'avantage de rendre possible de lire y compris le caractère RETOUR DE CHARIOT. Il existe cependant dans le DOS une routine DISC IN CHAR que vous pouvez utiliser à cet effet (voir également le listing du DOS) mais uniquement en langage machine. Vous pouvez ainsi écrire une routine adaptée à votre problème qui transmette le caractère lu à votre programme Basic.

Voici une petite routine qui simule en Basic l'instruction GET. Lorsque vous appelez cette routine, le fichier doit être ouvert en lecture.

```
10000 REM ==============================
10010 REM       ROUTINE  GET
10020 REM ==============================
10030 :
10040 IF LEN(in$)=0 THEN 10080
10050 ch$=LEFT$(in$,1)
10060 in$=MID$(in$,2)
10070 RETURN
10080 IF EOF THEN CLOSEIN:ch$="":RETURN
10090 LINE INPUT#9,in$
10100 IF NOT EOF THEN in$=in$+CHR$(13)
10110 GOTO 10050
```

Le caractère lu vous est transmis dans la variable ch$, le retour de chariot est maintenant lui aussi un caractère autorisé. Lorsque le dernier caractère a été lu, ch$ est vide et LEN(ch$) vaut donc zéro.

Si vous voulez lire à l'écran des valeurs numériques, vous utilisez le plus souvent l'instruction:

```
10 INPUT valeur
```

Si vous entrez maintenant un caractère qui n'est pas un nombre, le système répond par:

?Redo from start

AMSDOS ne peut bien sûr pas sortir un tel message d'erreur. Certains systèmes envoient dans ce cas une FILE TYPE ERROR, ce qui signifie: une chaîne de caractères a été transmise au lieu d'une valeur numérique. AMSDOS ne sort aucun message d'erreur mais effectue DE FACON INTERNE le calcul suivant:

```
10 INPUT#9,a$:valeur=val(a$)
```

On voit nettement ici pourquoi le caractère espace fait également office de symbole de séparation pour les nombres. En outre, chaque caractère non numérique fait office de symbole de séparation, à l'exception de "E" ou "e" qui est utilisé pour l'écriture exponentielle des nombres.

CHAPITRE 2: PROGRAMMATION AVANCEE DE LA DISQUETTE

2.1 LES VECTEURS DU DOS

Après que les chapitres précédents vous aient décrits de façon détaillée les possibilités d'AMSDOS et de CP/M pour une utilisation "normale", vous découvrirez dans les chapitres suivants un certain nombre de notions sur le mode de travail du DOS et sur l'utilisation du DOS en langage machine. Les chapitres suivants s'adressent donc essentiellement aux lecteurs qui disposent de certaines connaissances de base sur la programmation en langage machine du Z80. Si ces connaissances vous manquent, un certain nombre de notions développées dans les pages suivantes vous sembleront incompréhensibles. Mais n'abandonnez pas pour autant la lecture de cet ouvrage. Même si un certain nombre de choses vous semblent obscures au départ, vous pourrez néammoins utiliser beaucoup des informations qui vous sont fournies dans les pages suivantes.

On peut distinguer trois niveaux principaux de programmation en langage machine du lecteur de disquette du CPC. Le niveau supérieur de programmation est le plus simple à comprendre. A ce niveau, toutes les tâches essentielles sont accomplies par le DOS. Il s'agit au fond des équivalents en langage machine des instructions Basic que vous connaissez déjà depuis le chapitre précédent.
Le second niveau de programmation en langage machine de la disquette est un niveau qui serait impensable sans un listing de la ROM bien commenté. A ce niveau on dispose de routines dont certaines sont très simples mais dont d'autres aussi sont très complexes. Ces routines vont de la simple mise en marche du moteur du lecteur de disquette à la lecture et à l'écriture de données sur la disquette. Nous décrirons les plus importantes routines du DOS car on peut grâce à elles réaliser un certain nombre de choses très intéressantes qui sont impossibles à réaliser à partir du Basic.
La troisième possibilité de programmation du lecteur de disquette représente ce qu'on pourrait qualifier de programmation 'à pied'. Seuls quelques spécialistes se risqueront à ce niveau car il faut alors programmer directement les fonctions du disc controller. Mais comme les fonctions essentielles se trouvent déjà sous une forme ou sous une autre dans le DOS, cette programmation peut se limiter à quelques cas particuliers. Nous évoquerons également ce niveau de programmation et nous décrirons largement les possibilités du disc controller.

Mais commençons par les choses les plus simples. Ni le système d'exploitation du CPC 464 ni celui du CPC 664 ne sont conçus pour commander un lecteur de disquette. Le seul moyen de sauvegarde externe des données qu'ils connaissent est le lecteur de cassette. Ce n'est qu'avec la ROM AMSDOS que la commande du lecteur de disquette devient possible. Si cette ROM n'est pas disponible lors de la mise sous tension de l'ordinateur, les instructions LOAD, SAVE, OPENIN, OPENOUT, CAT et les instructions de manipulation des fichiers s'appliqueront au lecteur de cassette. Si par contre la ROM AMSDOS est disponible lors de l'initialisation du CPC, la situation change totalement et presque toutes les instructions qui s'appliquaient auparavant au lecteur de cassette agissent maintenant sur le lecteur de disquette. La seule exception est l'instruction SPEED WRITE puisque celle-ci n'a aucun effet sur le lecteur de disquette.

Le principe des vecteurs qui a été utilisé par les développeurs du CPC revêt un aspect essentiel pour la souplesse d'utilisation. Au lieu d'appeler directement l'action, c'est-à-dire la routine du système d'exploitation voulue, cette routine est appelée à travers une adresse de la RAM à laquelle figure un saut à la routine système correspondante. Le fait que la routine soit appelée à travers un vecteur situé en RAM peut sembler inutile au premier abord. En réalité cependant, ce principe garantit la compatibilité entre différentes versions de système d'exploitation. Ce n'est qu'à travers ce détour qu'il devient possible que des programmes machine puissent tourner aussi bien sur le 464 que sur le 664 bien que les deux ordinateurs possèdent des systèmes d'exploitation très différents à certains égards. Tant que les vecteurs figurent aux mêmes emplacements de la RAM et que les routines correspondantes remplissent des fonctions identiques, on peut entreprendre pratiquement n'importe quelles modifications du système d'exploitation. La compatibilité reste garantie. Un autre point très important est qu'on a en tant que programmeur toutes les cartes en main. Il n'est certes pas possible de modifier les routines de la ROM mais comme les vecteurs figurent en RAM, ils peuvent être sans problème 'détournés' sur les routines propres de l'utilisateur. Il est ainsi possible de modifier si nécessaire toutes les fonctions appelées à travers les vecteurs.
C'est exactement ce qui se produit lorsque le lecteur de disquette est connecté. Dans le CPC, 22 vecteurs sont prévus pour commander le lecteur de cassette. 13 de ces vecteurs sont modifiés pour le travail avec la disquette, c'est-à-dire que les adresses des routines à appeler sont modifiées. A travers ces 13 vecteurs sont appelées à partir du Basic

toutes les fonctions du lecteur de disquette. Mais ces vecteurs peuvent être également parfaitement utilisés en langage machine. Voici maintenant une présentation des vecteurs modifiés. Les noms employés correspondent à ceux du CPC 464 Firmware Manual.

&BC77	CAS IN OPEN	ouvre un fichier en lecture
&BC7A	CAS IN CLOSE	ferme correctement un fichier ouvert en lecture
&BC7D	CAS IN ABANDON	ferme immédiatement un fichier ouvert en lecture
&BC80	CAS IN CHAR	retire un caractère d'un fichier d'entrée
&BC83	CAS IN DIRECT	lit et place entièrement dans la mémoire un fichier d'entrée
&BC86	CAS RETURN	le dernier caractère est réécrit
&BC89	CAS TEST EOF	teste si la fin du fichier est atteinte
&BC8C	CAS OUT OPEN	ouvre un fichier en écriture
&BC8F	CAS OUT CLOSE	ferme correctement un fichier ouvert en écriture
&BC92	CAS OUT ABANDON	ferme immédiatement un fichier ouvert en écriture
&BC95	CAS OUT CHAR	écrit un caractère dans un fichier de sortie
&BC98	CAS OUT DIRECT	écrit entièrement une zone de la mémoire dans un fichier
&BC9B	CAS CATALOG	fonction identique à celle de l'instruction Basic CAT

Une fois que vous aurez lu cette table, nous vous conseillons d'oublier bien vite les noms utilisés ici. Il nous semble en effet vraiment illogique de désigner par CAS des routines utilisées avec le lecteur de disquette. Nous remplacerons toujours par la suite CAS par DISK. Le nom correspond ainsi à la fonction.

2.1.1 DISK IN OPEN &BC77

Avant que des données puissent être lues dans un fichier, ce fichier doit être ouvert. L'ouverture d'un fichier est prise en charge par la routine DISK IN OPEN. Il est à cet égard indifférent que le fichier à lire soit un fichier ASCII ou un programme. Cette routine doit être de toute façon appelée une fois.
Pour lire le fichier, certains paramètres sont bien sûr attendus. Le paramètre le plus important est le nom du fichier. Avec toutes les routines DISK, des paramètres sont transmis à travers les registres. Mais comme un nom de fichier se compose de 8 caractères plus 3, le nom ne peut

être entièrement transmis à travers les registres. Le registre double reçoit donc l'adresse à laquelle le nom de fichier figure en mémoire. Le nom de fichier peut se trouver n'importe où dans la RAM, y compris dans la partie de celle-ci qui se chevauche avec la ROM, par exemple dans les 16 premiers K de la mémoire du CPC.
Un autre paramètre, qui doit lui être transmis à la routine à travers le registre B, est constitué par la longueur du nom de fichier. Cela est une propriété très agréable de DISK IN OPEN puisque vous n'êtes pas ainsi obligé d'amener vous même le nom de fichier à une longueur prédéterminée. Vous pouvez ainsi entrer des noms de fichier ayant par exemple 4 caractères pour le nom et 2 pour l'extension (le type de fichier). La routine complètera elle-même le nom ainsi fourni de façon à ce qu'il atteigne la longueur requise. Mais n'oubliez jamais d'indiquer également la longueur de l'extension.
Un troisième paramètre est attendu par la routine DISK IN OPEN dans le registre double DE. Vous devez fournir ici l'adresse d'un buffer long de 2048 octets. Les données à lire seront écrites dans ce buffer. Il n'y a pas ici non plus de limites concernant la situation de ce buffer dans la mémoire.
Après que les trois paramètres aient été transmis aux registres que nous avons indiqués, la routine peut être appelée. Voyons un exemple d'un tel appel. Nous allons ouvrir le fichier 'TEST.DAT' en lecture. Programmée en assembleur, la séquence d'instructions se présente ainsi:

```
100          LD     HL,FILNAM          ;adresse du nom
110          LD     B,BUFF-FILNAM      ;longueur du nom
120          LD     DE,BUFF            ;adresse du buffer
130          CALL   DISK-IN-OPEN
140          RET
150 FILNAM:  DEFM   'test.dat'         ;nom du fichier
160 BUFF:    DEFS   &0800              ;place pour 2K
```

Mais que fait exactement cette routine et quels résultats produit-elle? Elle contrôle tout d'abord si le nom de fichier correspond à certaines règles formelles. C'est ainsi que certains caractères sont interdits dans le nom de fichier. Si un de ces caractères est rencontré, l'instruction est interrompue. Le nombre de caractères dans le nom de fichier ne doit pas non plus être supérieur à 15. Ce nombre comprend le numéro user, le numéro de drive avec le double point, 8 caractères pour le nom de fichier, le point de séparation et trois caractères d'extension. Un tel nom de fichier pourrait être par exemple: "0A:FILENAME.DAT".
En même temps que le nom de fichier est contrôlé, toutes les lettres

minuscules sont changées en majuscules. Les noms de fichiers plus courts sont complétés jusqu'à la longueur requise avec des caractères espace.
Après ce contrôle du nom de fichier, on examine s'il existe bien sur la disquette un fichier portant ce nom. S'il n'y a pas sur la disquette de fichier portant ce nom, par exemple parce que le nom a été mal ou incomplètement entré, AMSDOS annonce 'Filename not found'. C'est également ce message d'erreur que vous obtenez lorsque vous voulez charger un programme qui ne figure pas sur la disquette qui a été placée dans le lecteur.
Indépendamment du fait que le fichier figure ou ne figure pas sur la disquette, vous obtenez en tout cas dans les registres, après avoir appelé la routine, un certain nombre de paramètres importants en retour.
Tout d'abord l'état des flags Carry et Zéro indique si le DISK IN OPEN a réussi. A cet effet, le flag Carry est mis et le flag Zéro est annulé si l'OPEN a réussi. Si par contre le Carry et le Zéro sont tous deux annulés, c'est que vous avez tenté d'ouvrir en lecture un deuxième fichier. Vous aviez donc déjà ouvert un fichier. La troisième possibilité est que le flag Zéro soit mis mais que le Carry soit annulé. Dans ce cas, le fichier indiqué n'a pas été trouvé.
Si l'OPEN a réussi, la valeur de l'accumulateur permet de déterminer le type de fichier. S'il s'agit par exemple d'un programme Basic, l'accumulateur contiendra après la routine OPEN la valeur 0. Une valeur de &16 signifie que le fichier ouvert est un fichier ASCII. Vous trouverez les autres valeurs possibles, sous forme de table, sous DISK OUT OPEN.
Le double registre HL contient après ouverture du fichier l'adresse du header de fichier. La notion de header de fichier ne vous est certainement pas encore familière, c'est pourquoi nous allons l'expliquer brièvement ici. Mais c'est sous DISK OUT OPEN que vous trouverez une description détaillée de la structure du header de fichier. Le header de fichier contient une somme d'informations sur le fichier correspondant. C'est ainsi qu'il contient le nom du fichier, le type de fichier, la longueur du fichier ainsi que l'adresse à partir de laquelle le fichier a été écrit. Presque tous les fichiers qui sont sauvegardés sur la disquette, donc aussi les programmes Basic et les programmes machine, sont sauvegardés avec ces informations. La seule exception est constituée par les purs fichiers ASCII pour lesquels le header n'est pas sauvegardé avec le fichier mais produit par AMSDOS. Vous obtenez une autre information dans le registre double DE. Ici vous est communiquée l'adresse de départ du fichier. Cette information n'a bien sûr de sens que s'il s'agit d'un fichier de programme. Si le fichier ouvert est un programme Basic, la valeur contenue par DE est normalement de &0170. Pour

les programmes machine, DE contient l'adresse à partir de laquelle le programme a été écrit.
Enfin le registre double BC vous fournit comme dernière information la longueur du fichier. Il faut noter ici à nouveau que cette information n'est pas fournie pour les purs fichiers ASCII. Les fichiers ASCII peuvent être notablement plus longs que les 64K qu'un registre double permet seuls de représenter. Cette information n'a d'ailleurs de sens que pour le chargement des programmes.

2.1.2 DISK IN CLOSE &BC7A

Après que cette routine ait été appelé, un fichier ouvert en lecture sera fermé. Il n'est plus possible de lire dans ce fichier. DISK IN CLOSE est essentiellement requis lorsqu'il s'agit de lire des données dans un deuxième fichier. On ferme alors d'abord le fichier ouvert avec DISK IN CLOSE avant d'ouvrir en lecture le nouveau fichier avec DISK IN OPEN.
Si vous voulez absolument le savoir, vous pouvez déterminer, après avoir appelé cette routine et d'après le flag Carry, si un fichier était bien ouvert en lecture. Si le Carry est mis, c'est qu'un fichier était ouvert. Cette routine ne nécessite pas d'autres paramètres puisque pour le moment il ne peut jamais y avoir qu'un seul fichier d'entrée ouvert.

2.1.3 DISK IN ABANDON &BC7D

Cette routine remplit presque la même tâche que DISK IN CLOSE. Après DISK IN ABANDON aussi, un fichier d'entrée ouvert sera fermé. Cette routine est essentiellement conçue pour fermer le fichier en cas d'erreur. DISK IN ABANDON est également appelée par l'interpréteur Basic avant tout LOAD, SAVE ou CAT. Cela signifie que les fichiers ouverts sont toujours fermés après exécution d'une de ces instructions.

2.1.4 DISK IN CHAR &BC80

AMSDOS connaît fondamentalement deux méthodes différentes pour lire dans un fichier. La première méthode consiste à appeler la routine DISK IN CHAR, qui retire un caractère (CHARacter) du fichier. Cette routine ne nécessite pas non plus de paramètres puisqu'un seul fichier d'entrée peut être ouvert.
Le caractère lu dans le fichier est transmis à travers l'accumulateur. En

outre, les flags Carry et Zéro signalent si la routine a été exécutée correctement. Si un caractère a bien été retiré du fichier, le Carry est mis alors que le flag Zéro est annulé. Si vous essayez cependant de lire un caractère d'un fichier non ouvert, vous obtiendrez comme marque d'erreur les états Carry annulé, Zéro annulé. En outre le message d'erreur &0E figurera dans l'accumulateur. Cette marque indique que le fichier d'entrée n'était pas ouvert comme on s'y attendait.
D'autre part, si vous êtes arrivé à la fin du fichier, les flags seront tous deux annulés comme précédemment mais c'est la valeur &1A, la marque de END OF FILE, que vous obtiendrez dans l'accumulateur.
Les messages d'erreur possibles vous sont présentés à la fin du présent chapitre.
La lecture de données avec DISK IN CHAR ne peut s'effectuer que de façon séquentielle. Si vous avez par exemple lu, le 100ème caractère d'un fichier mais que vous vouliez accéder à nouveau au dixième caractère, il vous faut fermer le fichier puis le réouvrir.

2.1.5 DISK IN DIRECT &BC83

Comme nous l'avons déjà indiqué, AMSDOS offre deux possibilités de lecture des fichiers. Vous connaissez déjà la première méthode avec la routine DISK IN CHAR. DISK IN DIRECT représente la deuxième forme possible d'accès aux données d'un fichier. Contrairement à DISK IN CHAR, cette routine permet de lire et de placer dans la mémoire un fichier entier. L'application principale de cette routine est bien sûr le chargement de zones de la mémoire sauvegardées auparavant. C'est à travers cette routine que sont chargés les programmes Basic et les programmes machine.
Contrairement à DISK IN CHAR, DISK IN DIRECT exige un paramètre. Il s'agit de l'adresse de chargement du bloc de données. Cette adresse doit être transmise à travers le registre HL.

```
100            LD   HL,BUFFER      ;adresse de départ pour les données
110            CALL DISK IN DIRECT
120            RET
130 BUFFER     EQU  $              ;c'est ici que sont placées les
                                    données lues
```

Après appel de DISK IN DIRECT les flags indiquent de la façon qui nous est déjà familière si la routine a été ou non couronnée de succès. Carry mis/Zéro annulé signifie ici aussi que les données ont été lues

correctement. Les messages d'erreur possibles sont les mêmes que pour DISK IN CHAR. On obtient en outre dans le registre double HL, après appel de cette routine, ce qu'on appelle l'adresse ENTRY, fournie par le header de fichier. Veuillez voir sous DISK OUT OPEN quelle est la signification de cette adresse.

Il faut encore relever un certain nombre de particularités concernant les deux possibilités d'accès aux données d'un fichier. D'une part l'utilisation de ces deux routines est entièrement libre. Contrairement à ce qui est le cas en Basic, il est donc possible de lire aussi un programme avec DISK IN CHAR. Mais si un caractère d'un fichier d'entrée a déjà été lu avec DISK IN CHAR, le reste ne peut plus être lu et placé dans la mémoire avec la routine DIRECT. Inversement il n'est pas possible de continuer avec la routine CHAR la lecture d'un fichier commencée avec la routine DIRECT, sans compter qu'après DIRECT on a de toute façon atteint la fin du fichier. Il est d'autre part déconseillé de lire une seconde fois un fichier qui a été lu avec DISK IN DIRECT. Vous risqueriez en effet de détruire les données en mémoire.

2.1.6 DISK RETURN &BC86

Nous avons dit pour DISK IN CHAR qu'un fichier ne peut être traité que de façon séquentielle et cette affirmation est juste en principe. Avec DISK IN CHAR il est cependant possible de lire des caractères plus d'une fois. Avec DISK RETURN le caractère lu en dernier est réécrit dans le buffer et est à nouveau disponible pour la lecture. Si les 20 premiers caractères du fichier ont été lus, après avoir appelé 20 fois DISK RETURN, le prochain DISK IN CHAR lira à nouveau le premier caractère. Cette possibilité a toutefois ses limites. Le retour en arrière n'a été en effet prévu à l'origine que pour un seul caractère.
Un exemple illustrera où se situent les limites de cette routine. Dans cet exemple nous imaginerons un fichier de 4000 caractères. Après avoir lu 2048 caractères avec DISK IN CHAR, le buffer que nous avons défini avec DISK IN OPEN est vide, nous avons lu tous les caractères qu'il contenait. Lors de l'accès au prochain caractère, le 2049ème, le buffer sera rempli de nouvelles données à partir de la disquette. Si nous avons maintenant lu ce 2049ème caractère et que nous appelions la routine DISK IN RETURN plus d'une fois, il faudrait normalement que les 2048 premiers octets soient à nouveau chargé dans le buffer. Ce n'est cependant pas ce qui se produit. Au lieu de cela, le pointeur sur le prochain caractère à lire quitte la zone du buffer, de sorte que les caractères lus par la suite n'ont plus rien à voir avec le contenu du fichier.

DISK RETURN ne doit donc comme vous le voyez être utilisé que de façon très limitée. L'application principale de cette routine est le contrôle sélectif d'un caractère isolé. C'est ainsi que DISK RETURN est également utilisé pour tester le critère EOF &1A.

2.1.7 DISK TEST EOF &BC89

Cette routine vous permet de déterminer si vous avez atteint la fin du fichier. Le prochain caractère du fichier est lu à cet effet avec DISK IN CHAR et on teste s'il vaut &1A, soit 26 en décimal. Si la valeur du caractère lu n'est pas 26, le caractère est ensuite réécrit dans le buffer avec DISK RETURN et la routine renvoie un flag Carry mis et un flag Zéro annulé. Si par contre la valeur &1A a été trouvée, les flags Carry et Zéro sont annulés.
Cette routine ne peut être utilisée que lors d'une lecture caractère par caractère d'un fichier car cette routine utilise elle-même la routine DISK IN CHAR. Or l'utilisation de cette dernière routine est interdite

avec DISK IN DIRECT et entraîne des messages d'erreur.

2.1.8 DISK OUT OPEN &BC8C

Toutes les routines décrites précédemment présupposent que des données se trouvent déjà sur la disquette. Les descriptions de routines suivantes vous montrent maintenant comment vous pouvez sauvegarder des données et programmes sur la disquette en langage machine. De même que pour la lecture de données, le fichier voulu doit également être ouvert avant toute opération d'écriture. L'ouverture se fait en appelant la routine DISK OUT OPEN. De même que pour la lecture, différents paramètres doivent être transmis à la routine.
Il y a tout d'abord le nom de fichier sous lequel les données pourront ensuite être retrouvées. L'adresse du nom voulu est transmise à travers le registre double HL. La longueur du nom et l'adresse d'un buffer de 2048 octets doivent être transmis à travers le registre B et à travers le registre double DE, comme pour DISK IN OPEN.

```
100          LD    HL,FILNAM       ;adresse du nom
110          LD    B,BUFF-FILNAM   ;longueur du nom
120          LD    DE,BUFF         ;adresse du buffer
130          CALL  DISK-OUT-OPEN
140          RET
150 FILNAM:  DEFM  'test.dat'      ;nom du fichier
160 BUFF:    DEFS  &0800           ;place pour 2048 octets
```

La validité du nom de fichier est d'abord contrôlée et le nom est complété si nécessaire pour comporter le nombre correct de caractères. L'extension est ensuite remplacée par trois caractères dollar et ce nom est alors cherché sur la disquette. Sous ce nom est en effet d'abord créé un fichier provisoire dans lequel l'extension originale ne sera entrée que plus tard. Un header de fichier est d'autre part créé dont l'adresse est mise après exécution de la routine à la disposition de l'utilisateur dans le registre double HL. Cela suppose toutefois qu'aucune erreur ne se produise dans le DISK OUT OPEN. Vous pouvez tester ce fait comme toujours en examinant l'état du flag Carry. Si après exécution de cette routine le flag Carry est mis, c'est que tout est en ordre et que le fichier est ouvert en écriture. Si le Carry est par contre annulé, c'est qu'une erreur quelconque s'est produite. Les erreurs possibles sont par exemple des entrées incorrectes pour le nom de fichier. Les erreurs de lecture lors de l'examen du catalogue sont également annoncées comme erreurs.

Dans ce cas le contenu de HL est indéterminé.
Mais considérons le cas où le fichier de sortie a été ouvert correctement. Nous recevons dans ce cas l'adresse du header de fichier dans le registre double HL. Nous allons maintenant examiner ce header de fichier un peu plus précisément.
Le fichier n'est en fait rien d'autre qu'une zone de mémoire de 64 octets. Ces 64 octets contiennent les données les plus importantes sur le fichier. La signification des différents octets du header sont identiques pour la lecture et l'écriture des données. La structure du header est par ailleurs identique à la structure du header pour le travail sur cassette.
Les 16 premiers octets contiennent le nom de fichier. Un nom de fichier peut en effet comporter pour le travail sur cassette jusqu'à 16 caractères mais par contre pour le travail sur disquette, le nom ne se compose au maximum que de 8 caractères. Si on ajoute encore l'extension avec ses trois caractères maximum, nous n'avons toujours que 11 caractères. Comme premier caractère est en outre entré le numéro user. Cela donne un nombre de 12 octets et les 4 derniers caractères du nom de fichier dans le header fichier valent toujours 0.
Les octets 16 et 17 sont sans signification pour le lecteur de disquette. Ils contiennent pour le travail sur cassette le numéro de bloc actuel et une marque indiquant si le bloc actuel est le dernier bloc du fichier. Du fait d'une structure totalement différente, ces indications disparaissent sous AMSDOS.
L'octet suivant, numéro 18, est cependant également très important sous AMSDOS. Il contient le type de fichier. Le type de fichier est codé sous forme binaire, c'est-à-dire que les différents bits de cet octet ont une signification déterminée.
Le bit 0 indique si le fichier est protégé. Pour les programmes qui ont été sauvegardés avec SAVE"nom de fichier",P, le bit 0 du type de fichier est mis.
Les bits 1, 2 et 3 déterminent le type de fichier proprement dit. Si les trois bits sont annulés, il s'agit d'un programme Basic. Si le bit 1 est mis, le fichier est un fichier binaire, donc une zone de mémoire du CPC. Le type de fichier ASCII est produit par la mise des bits 1 et 2.
Les bits 4 à 7 ne sont normalement pas mis et ils représentent, selon le Firmware Manual du CPC, le numéro de version, sans que cette notion apparaisse très claire. Seuls les fichiers ASCII ont le numéro de version 1, le bit 4 du type de fichier étant mis.
Après un DISK OUT OPEN réussi, le type de fichier est mis sur fichier ASCII. C'est à vous d'entrer ici la valeur correcte dont vous avez besoin.
Les octets 19 et 20 sont encore un reste du travail sur cassette. Ils

contiennent le nombre d'octets du bloc de données actuel mais ils sont sans signification pour le travail sur disquette.
Plus importants sont les deux octets suivants numéros 21 et 22. Ils contiennent l'adresse à partir de laquelle les données ont été écrites. Cette indication n'est bien sûr pas encore fournie pour DISK OUT OPEN. Ce n'est que lors de l'écriture dans un fichier avec DISK OUT DIRECT (voir ci-dessous) que ce champ du header de fichier est rempli. Pour les fichiers ASCII vous ne trouvez ici que la valeur 0 car ce paramètre n'a aucun sens dans ce cas.
L'octet 23 du header de fichier vaudra toujours &FF. Pour le travail sur cassette, cette valeur indique que le bloc de données actuellement lu est le premier du fichier. Cette indication n'a pas non plus de sens pour le travail sur disquette.
Les octets 24 et 25 contiennent la longueur du fichier. Cette valeur est transmise dans le registre double DE après DISK IN OPEN. Pour les fichiers ASCII, cette valeur est toujours 0 car les fichiers ASCII ne voient leur taille limitée que par la capacité de la disquette. Celle-ci excède toutefois la valeur maximale pouvant être représentée avec deux octets.
Les derniers octets utilisés, 26 et 27 contiennent sur les programmes machine l'adresse de départ du programme. Lorsque vous sauvegardez une zone de la mémoire avec l'instruction SAVE"memoire.bin",b,1000,2000,1200, la valeur 1200 sera alors placée dans ces deux cases mémoire en format hexadécimal. Si vous travaillez cependant avec les mêmes routines en langage machine, vous devez entrer cette valeur vous même.
Tous les autres octets du header de fichier ne sont utilisés d'aucune manière par AMSDOS. Si vous en avez l'utilité, vous pouvez les employer. Après DISK OUT OPEN, ces octets sont mis sur 0 et sont ensuite à votre libre disposition.

2.1.9 DISK OUT CLOSE &BC8F

Un fichier de sortie doit également être fermé. Mais la nécessité de fermer un fichier est beaucoup plus grande lors de l'écriture que lors de la lecture. Cela vient du fait que chaque caractère n'est pas écrit directement sur la disquette mais d'abord dans le buffer de 2048 octets qui a été créé avec DISK OUT OPEN. Si ce buffer déborde, c'est-à-dire si le nombre de caractères à stocker est plus grand que 2048, ce buffer est écrit sur la disquette. Lors de la fermeture d'un fichier, il peut donc y avoir dans le buffer jusqu'à 2047 caractères. Avec l'appel de la routine CLOSE, ce reste figurant dans le buffer est également écrit sur la

disquette. Ce n'est qu'à partir de ce moment que le fichier sur la disquette sera complet. Vous risquez donc de perdre, dans l'hypothèse la moins favorable, 2047 octets si vous changez la disquette alors qu'un fichier de sortie est encore ouvert.
D'autre part, le nom de fichier provisoire (avec l'extension .$$$) est remplacé par le nom de fichier original. Un fichier existant éventuellement déjà sous ce nom verra alors son nom changer, il recevra l'extension .BAK. Si vous voyez dans le catalogue un fichier ayant cette extension, il s'agit en général d'un fichier qui n'a pas été refermé correctement.
La routine CLOSE ne nécessite pas la transmission de paramètres particuliers car, comme pour la lecture, un seul fichier peut être ouvert à la fois. Après exécution de cette routine, vous pouvez déterminer, au vu de l'état des flags Carry et Zéro, si le CLOSE a été exécuté correctement. Dans ce cas, le Carry est à nouveau mis alors que le flag Zéro est annulé.

2.1.10 DISK OUT ABANDON &BC92

Si une erreur grave se produit lors de l'écriture dans un fichier, le fichier peut être fermé grâce à cette routine. Un exemple d'une telle erreur grave serait un message indiquant que la disquette est pleine. Dans ce cas vous devriez appeler DISK OUT ABANDON car les blocs occupés jusqu'ici seront alors libérés sur la disquette. De même, le nom du fichier n'apparaîtra pas dans le catalogue. Dans un tel cas, toutes les données écrites jusqu'à présent devront être écrites à nouveau sur une disquette comportant plus de place mémoire libre.

2.1.11 DISK OUT CHAR &BC95

Comme pour la lecture de données, AMSDOS connaît deux moyens différents pour écrire des données sur la disquette. La première méthode passe par DISK OUT CHAR. Cette routine écrit le caractère figurant dans l'accumulateur dans le buffer OPENOUT. Si un débordement se produit, les caractères écrits jusqu'ici dans le buffer sont sauvés sur la disquette et le buffer est ainsi à nouveau libéré.
Cette routine permet en principe d'écrire n'importe quel caractère dans un fichier. Si le fichier est cependant plus tard lu avec DISK IN CHAR, vous devrez alors être très prudent dans l'utilisation du caractère &1A (26 en décimal). Ce caractère risquerait en effet sinon d'être interprété

lors d'une lecture ultérieure comme la marque EOF, même si le fichier ne se termine pas du tout à cet endroit.
Hormis le caractère à écrire qui doit être placé dans l'accumulateur, aucun autre paramètre ne doit être transmis à cette routine. Comme pour les autres routines, après exécution de DISK OUT CHAR, le Carry est mis si le caractère a été écrit correctement. Si par contre les flags Carry et Zéro sont annulés, c'est que vous avez négligé d'ouvrir correctement le fichier de sortie nécessaire.

2.1.12 DISK OUT DIRECT &BC98

Cette routine représente la seconde possibilité pour AMSDOS de sauvegarder des données dans un fichier sur disquette. Au contraire de l'écriture de caractères isolés, cette routine est utilisée pour sauvegarder des zones entières de la mémoire du CPC. Il faut pour cela transmettre différents paramètres à DISK OUT DIRECT.
Le registre double HL doit d'abord recevoir l'adresse de départ de la zone de mémoire à écrire. A partir de cette adresse, les données seront écrites dans le fichier.
La prochaine valeur importante est la taille de fichier voulue. Vous devez à cet effet charger dans le registre double DE le nombre d'octets à sauvegarder. Il est donc évident que la taille maximale d'un tel fichier est 64K ou 65536 octets. Des fichiers de taille plus importante ne seraient d'ailleurs pas utiles. Cette grandeur de fichier suffit pour sauvegarder toute la mémoire RAM du CPC sur disquette, ce qui d'ailleurs n'est pas très intéressant puisqu'un tel fichier ne peut plus être rechargé entièrement.
Si vous le souhaitez, vous pouvez transmettre dans le registre double BC l'adresse ENTRY (par exemple l'adresse de départ pour des programmes machine). Le contenu de ce registre double est écrit dans les octets 26 et 27 du header de fichier.
Vous pouvez transmettre un dernier paramètre à travers l'accumulateur. Il s'agit du type de fichier voulu. Cette indication sera également écrite dans le header de fichier, dans l'octet 18 du header.

Comme pour la lecture de fichiers, il est également important en écriture de ne pas changer de méthode d'écriture. On ne peut donc pas sauter dans un fichier d'une méthode à l'autre. Une fois que vous avez écrit un caractère avec DISK OUT CHAR, toute tentative pour utiliser ensuite DISK OUT DIRECT entraînera un message d'erreur.
Il n'est malheureusement pas non plus possible d'utiliser dans un fichier

DISK OUT DIRECT plus d'une fois. Même si, après un autre DISK OUT DIRECT, AMSDOS envoie le message OK, il y aura, au plus tard lors de DISK OUT CLOSE, le message 'File not open as expected'. Le fichier ne sera pas non plus écrit correctement. Après que cette routine ait donc été exécuté une fois, le fichier de sortie doit être refermé. Une autre zone de mémoire du CPC ne pourra être sauvegardée sur la disquette que dans un autre fichier.

2.1.13 DISK CATALOG &BC9B

Ne serait-ce que pour cette propriété intéressante, la disquette doit ̂etre préférée à la cassette. Vous obtenez en un temps très bref un aperçu des fichiers figurant sur le lecteur de disquette. En outre, le nombre de blocs libres sur la disquette est affiché.
Avant d'appeler DISK CATALOG vous devez indiquer dans le registre double DE l'adresse de départ d'une mémoire buffer de 2048 octets. Ce buffer n'était pas au fond indispensable et il a été mis en place certainement en premier lieu pour des raisons de compatibilité avec le travail sur cassette. Mais les programmeurs de l'AMSDOS ont eu une idée très intéressante pour tirer profit de ce buffer. Lorsque vous appelez DISK CATALOG, les noms des fichiers sont en effet lus sur la disquette et placés dans ce buffer, avec l'information concernant le nombre de blocs occupés. Les noms de fichier ne sont cependant pas écrits simplement dans l'ordre dans lequel ils apparaissent sur la disquette mais ils sont triés d'après l'ordre alphabétique. C'est pourquoi vous obtenez toujours un catalogue aussi joliment trié lorsque vous entrez l'instruction CAT. Par contre, avec l'instruction DIR, les fichiers sont affichés dans l'ordre dans lequel ils figurent effectivement sur la disquette, c'est-à-dire non triés.

2.1.14 LE PATCHING (OU DETOURNEMENT) DES VECTEURS

Au début de ce chapitre nous avons expliqué en détail les avantages des vecteurs situés en RAM. Mais si vous voulez tirer parti d'un avantage essentiel, de la facilité de modification des routines correspondantes, il vaut mieux ne pas vous mettre au travail de façon trop précipitée. La médaille a en effet un revers dangereux.

Un regard sur le listing de la ROM au chapitre 3 montre où se situe la difficulté. Après initialisation d'AMSDOS, les 13 vecteurs présentent le

même contenu. Dans tous les cas, les trois octets de chaque vecteur sont:

```
RST    &18
DEFW   &A88B
```

Une simple boucle PEEK confirme cette affirmation. La routine RST &18, également appelée RST 3 permet d'appeler comme sous-programme, donc de sauter presque comme avec CALL à n'importe quelle routine dans n'importe quelle ROM ou dans la RAM. Pour cela cependant, l'adresse sur deux octets figurant après l'instruction Restart ne suffit pas. En effet on ne peut placer dans deux octets qu'une seule adresse, sans qu'il soit possible d'ajouter une valeur de sélection de la ROM. La solution de ce problème est la suivante: l'adresse derrière RST indique l'adresse de la mémoire à laquelle figure la véritable adresse sur trois octets.
Dans notre cas, il nous faut regarder en &A88B pour obtenir l'adresse et l'octet de sélection de la ROM pour la routine effective du DOS. A l'adresse &A88B figurent les valeurs &30, &CD et &07. Ceci donne en lecture 'en clair' l'adresse &CD30 dans la ROM d'extension 7.
Si l'on regarde maintenant à l'adresse indiquée de la ROM AMSDOS on trouve une routine assez particulière qui détermine l'adresse du vecteur appelé à travers une manipulation de la pile (en effet, sur la pile se trouve l'adresse du vecteur + 3). Une valeur de &10D2 est ajoutée à cette adresse et placée sur le haut de la pile. Un RETurn conduit alors à la routine demandée. Que s'est-il donc passé?

On comprend pourquoi on a choisi cette voie détournée pour appeler les différentes routines si l'on se représente que le RST 3 a le même effet qu'un CALL. Lors d'un CALL, une adresse de retour est placée sur la pile. Mais cette adresse pointe DERRIERE le vecteur appelé, sur la prochaine entrée du bloc de sauts. Notre programme doit cependant être poursuivi à partir de l'emplacement qui suit l'appel du vecteur. C'est pourquoi l'adresse de retour de RST doit absolument disparaître de la pile.
C'est exactement cette tâche qui incombe à la routine en &CD30. L'adresse de retour à écarter n'est cependant pas simplement jetée aux oubliettes. Elle est au contraire additionnée à la valeur &10D2, ce qui donne l'adresse effective de la routine à appeler. Si un RST vient sur l'adresse &CD30 à partir d'une autre adresse qu'une entrée du bloc de sauts, l'addition donne une valeur totalement insensée et un "plantage" ou un Reset peuvent en être les conséquences probables.

Mais assez d'explications. Voyons comment on peut malgré tout détourner ces routines.

L'exemple choisi est très simple. La seule fonction de ce détournement est de montrer comment un tel détournement fonctionne. Pour cet exemple, nous détournerons le vecteur CAT.

```
100 INIT:    LD    HL,&BC9C     ;adresse du vecteur CAT
110          LD    TEMP,HL      ;sauvegarder
120          LD    HL,PATCH     ;nouvelle adresse du RST
130          LD    &BC9C,HL     ;patcher
140          RET                ;c'est bon
150 PATCH:   EQU   $            ;ici pourrait figurer votre routine
160          LD    HL,TEMP      ;adresse originale
170          LD    &BC9C,HL     ;entrer à nouveau
180          CALL  &BC9B        ;vecteur CAT avec adresse correcte
190 PATCH1:  EQU   $            ;ici aussi pourrait être votre routine
200          CALL  INIT         ;restaurer pour le prochain appel
210          RET                ;tout  est fini
```

Comme vous le voyez, nous ne faisons vraiment rien d'autre que détourner le vecteur. Toutefois, de cette façon, vous pouvez intervenir aussi bien avant qu'après la routine.

Un exemple pratique de la méthode utilisée est la réparation de MERGE et de CHAIN MERGE. Vous pourriez désassembler ce petit programme pour voir comment l'intervention dans la routine DISK IN CHAR y est programmée.

2.2 LES EXTENSIONS D'INSTRUCTIONS DE L'AMSDOS

Les routines décrites dans le chapitre précédent sont loin d'épuiser toutes les possibilités qui s'offrent au programmeur. Les instructions étendues, marquées en Basic par un trait vertical (SHIFT-@), peuvent être également utilisées sans difficulté en langage machine. Voyons maintenant en détail comment procéder.

Vous connaissez déjà les instructions possibles:

CPM	A
DISC	B
DISC.IN	DRIVE
DISC.OUT	USER
TAPE	DIR
TAPE.IN	ERA
TAPE.OUT	REN

Au contraire des routines DISK décrites précédemment, il n'y a pas de vecteurs pour les instructions étendues, de sorte qu'il n'est pas possible de les appeler directement. Pour les extensions d'instructions, il convient de prendre certaines mesures particulières pour arriver à l'action voulue. Essayons donc de nous familiariser avec le mécanisme nécessaire.

2.2.1 PROGRAMMATION DES EXTENSIONS EN ASSEMBLEUR

Si les instructions d'une extension d'instruction quelconque doivent être exécutées, il faut tout d'abord que l'adresse de la routine soit connue. Comme ces routines peuvent se trouver aussi bien dans la RAM sous la forme de ce qu'on appelle une extension RSX que dans les ROMs dans les 16 K supérieurs du CPC, il faut donc en outre déterminer une éventuelle adresse de ROM. Une routine spéciale du Kernal du CPC sert à retrouver ces indications lorsque nous indiquons le nom de l'extension d'instruction voulue. Il faut à cet effet que l'adresse du nom soit entrée dans le registre double HL. Il ne faut d'autre part indiquer le nom qu'en majuscules. Le dernier caractère du nom doit être marqué par la mise du bit 7 de ce caractère. C'est exactement sous cette forme que figurent en ROM et en RAM tous les noms des extensions d'instructions.
Si le nom se trouve donc en RAM sous la forme voulue et que HL contient l'adresse de ce nom, il suffit d'appeler la routine KL FIND COMMAND.

KL FIND COMMAND est une routine du kernal du CPC qui a un vecteur en RAM à l'adresse &BCD4. Après que cette routine ait été exécutée, vous obtenez en retour les paramètres suivants:
Si le flag Carry n'est pas mis, c'est que l'instruction correspondante n'a pas été trouvée. Soit vous n'avez pas entré correctement le nom de l'extension d'instruction, soit l'extension n'est pas encore initialisée. Avec le lecteur de disquette, ce dernier cas peut survenir si vous allumez d'abord le CPC et ensuite seulement le lecteur de disquette. Mais si vous appuyez alors simultanément sur les trois touches CTRL, SHIFT et ESC, un Reset se produira au cours duquel l'extension d'instruction sera 'intégrée', de sorte qu'elle sera ensuite à votre disposition.
Si par contre le flag Carry est mis après la routine KL FIND COMMAND, vous obtenez en retour l'adresse de la routine voulue dans le registre double HL. Vous trouverez dans le registre C l'adresse ROM requise.

Le fait que ce soient précisément ces registres qui reçoivent ces informations résulte d'un choix délibéré des programmeurs du système d'exploitation. Il existe en effet une routine Kernal appelée KL FAR PCHL qui peut appeler comme sous-programme n'importe quelle adresse dans une ROM ou dans la RAM. L'adresse de la routine doit à cet effet être transmise dans HL et l'adresse ROM requise dans le registre C. C'est exactement ici que se trouvent les valeurs nécessaires après KL FIND COMMAND, de sorte que rien ne manque plus pour appeler la routine.

Examinons ce mécanisme en prenant l'exemple de l'instruction IDIR, qui ne nécessite pas de paramètres supplémentaires, avant que nous n'en venions aux instructions qui nécessitent des paramètres sous forme de variable numériques ou alphanumériques.

```
100            LD    HL,COMMAND         ;adresse de l'instruction
110            CALL  KL-FIND-COMMAND    ;adresse &BCD4
120            RET   NC                 ;pas trouvé instruction
130            XOR   A                  ;pas de paramètres
140            CALL  KL-FAR-PCHL        ;adresse &001B
150            RET
160  COMMAND:  DEFM  'DI','R'+&80       ;nom de l'instruction
```

Les deux routines du Kernal utilisées ici sont d'une efficacité vraiment fantastique. Plus personne n'a à se perdre dans de longues tables d'adresses des routines nécessaires. Il suffit simplement que le nom de la routine soit connu. Cette puissance n'a été à notre connaissance atteinte jusqu'ici sur aucun autre ordinateur domestique.

La programmation ne devient que légèrement plus complexe lorsque des paramètres doivent être transmis à l'extension d'instruction. Prenons d'abord le cas où seules des valeurs numériques entières doivent être transmises. C'est par exemple le cas pour l'instruction |USER. Les numéros USER autorisés sont les valeurs de 0 à 15. Voyons un programme d'exemple d'appel de l'instruction user avant de traiter des particularités de la transmission des paramètres:

```
100            LD    HL,COMMAND          ;adresse de l'instruction
110            CALL  KL-FIND-COMMAND     ;adresse &BCD4
120            RET   NC                  ;pas trouvé instruction
130            LD    A,1                 ;transmettre un paramètre
140            LD    IX,NUMBER           ;numéro user voulu
150            CALL  KL-FAR-PCHL         ;adresse &001B
160            RET
170  COMMAND:  DEFM  'USE','R'+&80       ;nom de l'instruction
180  NUMBER:   DEFW  0004
```

Vous voyez tout d'abord que nous transmettons dans l'accumulateur à la routine |USER le nombre de paramètres transmis. Toute autre valeur que 1 entraîne dans ce cas la sortie de 'Bad command'.
Le registre IX pointe sur le numéro user voulu, dans notre exemple le numéro user 4. Comme seules des valeurs 16 bits peuvent être transmises, la valeur de NUMBER doit être définie à la ligne 180 avec l'instruction DEFW, sous la forme d'une valeur sur deux octets (soit 16 bits).
La programmation n'est donc pas tellement plus compliquée dans ce cas. Il faut cependant un peu plus de travail lorsque des chaînes de caractères doivent être transmises à l'extension d'instruction. C'est par exemple le cas pour les instructions |ERA, |REN et |DRIVE.
Comme vous le savez déjà, les chaînes de caractères ne peuvent pas être transmises directement aux instructions de l'extension d'instruction. Au lieu de cela, c'est le pointeur de variable que vous obtenez avec la fonction arobas @ qui doit être transmis. Le pointeur de variable pointe sur ce qu'on appelle le string descriptor ou descripteur de chaîne. La description détaillée du descripteur de chaîne et de la gestion des chaînes de caractères en Basic sort du cadre de cet ouvrage.

Prenons d'abord l'exemple le plus simple où une seule chaîne doit être transmise comme paramètre, l'instruction |ERA. Dans notre exemple, nous supprimerons un fichier portant le nom 'ADRESSES.DAT'.

```
100            LD    HL,COMMAND          ;adresse de l'instruction
```

```
110             CALL  KL-FIND-COMMAND    ;adresse &BCD4
120             RET   NC                 ;pas trouvé instruction
130             LD    A,1                ;transmettre un paramètre
140             LD    IX,VARPTR          ;descripteur de variable
150             CALL  KL-FAR-PCHL        ;adresse &001B
160             RET
170  COMMAND:   DEFM  'ER','A'+&80       ;nom de l'instruction
180  VARPTR:    DEFW  DESCRIP            ;adresse du descripteur
190  DESCRIP:   DEFB  12                 ;longueur de la variable
200             DEFW  FILNAM             ;adresse du nom
210  FILNAM:    DEFM  'ADRESSES.DAT'     ;fichier à supprimer
```

Les choses se compliquent encore un peu plus lorsque deux chaînes doivent être transmises, comme c'est le cas pour l'instruction IREN. L'exemple suivant vous montre comment vous pouvez changer le nom d'un fichier 'ADRESSES.DAT' en 'ADRESSES.ANC'.

```
100             LD    HL,COMMAND         ;adresse de l'instruction
110             CALL  KL-FIND-COMMAND    ;adresse &BCD4
120             RET   NC                 ;pas trouvé instruction
130             LD    A,2                ;transmettre deux noms de fichier
140             LD    IX,VARPTR          ;descripteurs de variable
150             CALL  KL-FAR-PCHL        ;adresse &001B
160             RET
170  COMMAND:   DEFM  'RE','N'+&80       ;nom de l'instruction
180  VARPTR:    DEFW  DESCOLD            ;adresse descripteur ancien nom
190             DEFW  DESCNEW            ;adresse descripteur nouveau nom
200  DESCOLD:   DEFB  12                 ;longueur de la variable
210             DEFW  OLDNAME            ;adresse ancien nom de fichier
220  OLDNAME:   DEFM  'ADRESSES.DAT'     ;ancien nom de fichier
230  DESCNEW:   DEFB  12                 ;longueur de la variable
240             DEFW  NEWNAME            ;adresse nouveau nom de fichier
250  NEWNAME:   DEFM  'ADRESSES.ANC'     ;nouveau nom de fichier
```

Même si dans cet exemple toutes les données ont été bien disposées à la suite les unes des autres, elles peuvent cependant figurer en réalité n'importe où dans la mémoire du CPC. L'ordre ne joue pas non plus de rôle. Il faut simplement que les pointeurs sur les descripteurs figurent en mémoire dans le bon ordre et immédiatement les uns à la suite des autres.

Les connaissances acquises dans ce chapitre devraient maintenant vous

permettre de pouvoir utiliser également en langage machine toutes les possibilités disponibles à partir du Basic. Cela fait certainement déjà beaucoup mais cela ne représente qu'une partie de ce qui est effectivement possible. Le but de la programmation en langage machine est en effet justement de reculer les limites des possibilités existant en Basic. Cependant nous n'en sommes pas encore arrivé là.

Nous allons donc étudier dans le chapitre suivant quelles possibilités supplémentaires s'offrent au programmeur en langage machine.

2.2.2 LES INSTRUCTIONS 'CACHEES' DE L'EXTENSION D'INSTRUCTION

En étudiant le listing de la ROM nous avons découvert des possibilités qui ne sont nullement évoquées dans le manuel d'utilisation. Outre les 14 instructions connues de l'extension d'instructions, il y a encore 9 autres instructions très intéressantes qui peuvent être mises en oeuvre comme les instructions décrites jusqu'ici. Toutefois ces instructions ne peuvent pas être utilisées à partir du Basic comme par exemple l'instruction IDIR.
Toutes ces instructions ont un nom d'un caractère. Ces noms sont &01, &02 ... &09. Comme pour tous les noms de l'extension d'instruction le 7ème bit du dernier caractère doit être mis, les noms effectifs sont donc &81, &82 ... &89. De tels noms ne peuvent être entrés dans des programmes Basic. L'utilisation de ces instructions est donc en fait ainsi réservée au programmeur en langage machine. Pour certaines de ces instructions, leur utilisation en Basic ne présenterait d'ailleurs aucun intérêt comme vous le montrera la description des différentes instructions.

2.2.2.1 L'INSTRUCTION &81 MESSAGE ON/OFF

Cette instruction permet d'interdire les messages d'erreur qui peuvent se produire en liaison avec les instructions suivantes &82 à &89. Il s'agit en particulier des messages d'erreur du disk controller qui demandent à l'utilisateur d'entrer C, I ou R pour Chancel, Ignore et Retry. Pour arriver à cette interdiction des messages d'erreur, le programmeur doit placer dans l'accumulateur, avant appel de l'instruction, une valeur non nulle.

```
          LD    HL,COMMAND       ;adresse de l'instruction
          CALL  KL-FIND-COMMAND  ;adresse &BCD4
          RET   NC               ;pas trouvé instruction
          LD    A,&FF            ;valeur pour interdire les
                                 messages
          CALL  KL-FAR-PCHL      ;adresse &001B
          RET
COMMAND:  DEFB  &81              ;nom de l'instruction
```

La seule fonction de ce programme est de transférer la valeur dans l'accumulateur dans la case mémoire &BE78. Vous pouvez arriver certainement plus rapidement au même résultat en écrivant explicitement dans cette case mémoire. Mais si l'adresse de cette case mémoire devait

être modifiée, pour des raisons quelconques, dans des versions ultérieures de l'AMSDOS, la routine de l'AMSDOS que nous venons de décrire sera certainement adaptée de façon à ce que ce soit la nouvelle adresse qui soit fournie. Pour des raisons de compatibilité avec d'éventuelles versions ultérieures, il faut donc absolument que vous utilisiez la routine montrée ci-dessus, si vous ne programmez pas exclusivement pour vous même.

En Basic 1.0, cette routine ne peut être utilisée car vous devez vous en tenir aux messages d'erreur sur l'écran. Le numéro d'erreur transmis dans l'accumulateur ne vous est en effet pas fourni en Basic 1.0. Le seul effet serait d'interrompre le programme en cours et de faire apparaître sur l'écran le message laconique 'BREAK'. Avec le Basic 1.1 du CPC 664 vous pouvez par contre employer cette routine et interroger les erreurs avec ON ERROR GOTO et avec la variable système DERR. Pour les programmes Basic, le programme d'interception des messages d'erreur que vous trouverez plus tard dans la collection de programmes, est cependant plus adapté.

2.2.2.2 L'INSTRUCTION &82 PARAMETRES DE DRIVE

Cette instruction autorise la modification des données du lecteur de disquette telles que le délai d'attente entre la mise en marche du moteur et le moment où est atteint le nombre de rotations voulu, ou la constante correspondant au délai qu'il faut attendre après un changement de piste, ainsi que le temps de fin de rotation du moteur du disque. Les durées pour le HEAD LOAD TIME et pour le HEAD UNLOAD TIME peuvent être également fixées à nouveau avec cette routine.
Contrairement aux routines précédentes de l'extension d'instruction, &82 ne peut pas être appelée avec les routines connues jusqu'ici. &82 attend en effet dans le registre double HL le début de la table pour les données du lecteur de disquette. L'appel avec KL FAR PCHL n'a donc pas lieu d'être ici puisque cette routine attend en HL l'adresse de la routine à appeler. Il y a cependant également d'autres méthodes pour appeler une instruction d'extension.

```
100        LD   HL,COMMAND         ;adresse de l'instruction
110        CALL KL-FIND-COMMAND    ;adresse &BCD4
120        RET  NC                 ;pas trouvé instruction
130        LD   (FARADR),HL        ;ranger adresse de la routine
140        LD   A,C                ;sélection ROM dans l'accu
```

```
150            LD    (FARADR+2),A     ;et sauver également
160            LD    HL,NEWTAB        ;tables des paramètres disque
170            RST   &18              ;agit comme un CALL de la routine
                                      voulue
180            DEFW  FARADR           ;vecteur sur far adress 3 octets
190            RET
200  COMMAND:  DEFB  &82              ;nom de l'instruction
210  FARADR:   DEFS  3                ;place pour l'adresse 3 octets
220  NEWTAB:   DEFS  7                ;7 octets doivent être transmis à
                                      &82
230  HDUNLD:   DEFS  1                ;ici doit figurer l'adresse HEAD
                                      UNLOAD TIME voulue
240  HEADLD:   DEFS  1                ; HEAD LOAD TIME d'après fiche
                                      technique du FDC
```

Contrairement aux appels précédents de routines de l'AMSDOS, nous avons utilisé ici une instruction RESTART. Le RST représente une forme particulière de l'instruction CALL. Ce RST permet, comme avec la routine KL FAR PCHL, d'appeler n'importe quelle adresse dans n'importe quelle ROM possible ou dans la RAM. L'avantage de RST &18 est cependant qu'aucun registre n'est nécessaire. Nous pouvons donc avec cette routine transmettre aux sous-programmes à appeler des paramètres même dans le registre double HL et dans le registre C, ce qui n'était pas possible avec la routine du KERNAL utilisée jusqu'ici. Pour cela, les trois octets de la FAR adresse doivent être placés dans la mémoire du CPC. Ces trois octets, soit l'adresse sur deux octets et l'octet de sélection de la ROM, doivent absolument être placés dans la mémoire dans l'ordre indiqué et l'un après l'autre.

Nous allons maintenant examiner encore une fois comment la table requise doit se présenter.
La première entrée de la table est une valeur 16 bits qui définit le temps qu'il faut attendre après la mise en marche du moteur du disque. Il est évident que la disquette n'atteint pas la vitesse de rotation voulue immédiatement après que le moteur du lecteur ait été mis en marche. Cela ne se produit qu'au bout d'environ une seconde. La valeur standard après la mise sous tension est donc de 50 décimal. Cette valeur est décomptée avec un ticker event. Comme le ticker est exécuté tous les cinquantièmes de seconde, on obtient un délai d'attente d'exactement une seconde.
La deuxième entrée de la table décrit le temps de fin de rotation des moteurs après le dernier accès à la disquette. Ici aussi on travaille avec le ticker. Après initialisation d'AMSDOS, la valeur 16 bits utilisée

est 250 décimal. Cela correspond à une période de fin de rotation de 2,5 secondes.
La troisième entrée de la table est une valeur sur un octet. La valeur standard est ici &AF. Cette valeur n'est utilisée que par la routine &86, formatage d'une piste. Elle ne doit pas être modifiée.
La valeur 16 bits suivante définit également des délais d'attente. L'octet fort de cette valeur détermine le temps qu'il faut attendre après un changement de piste. C'est un délai d'attente de 12 ms qui est fixé comme valeur standard. Ce délai d'attente dépend du lecteur de disquette utilisé.
Deux autres délais d'attente doivent être définis dans cette table. Ce sont les valeurs pour HEAD LOAD TIME et HEAD UNLOAD TIME qui sont requises par le circuit intégré du controller. Les valeurs prescrites en ROM sont 32 ms pour HEAD UNLOAD TIME et 16 ms pour HEAD LOAD TIME. Vous trouverez plus loin la signification exacte de ces valeurs, dans la description du FDC 765.

L'instruction &82 est exécutée automatiquement par AMSDOS après la mise sous tension ou après un Reset. Sous CP/M également, cette fonction est exécutée une fois après le lancement de CP/M. Sous CP/M il est possible de définir comme vous le souhaitez les différents paramètres dans le programme SETUP.COM. Vous trouverez la table utilisée après le Reset à l'adresse &C5D4 dans la ROM de l'AMSDOS.

2.2.2.3 L'INSTRUCTION &83 PARAMETRES DE FORMAT DE DISQUE

Comme vous le savez, le CPC est prévu pour pouvoir travailler avec trois formats de disquette différents. L'instruction &83 permet de déterminer le format d'une disquette placée dans le lecteur de disquette actif. Pour chaque format figure une table dans la ROM de l'AMSDOS. Suivant la valeur transmise dans l'accumulateur, la table requise sera placée dans la RAM du CPC. L'utilisateur peut donc ainsi mettre en place les tables qui conviennent dans des programmes machine avec accès direct de secteur. En Basic, l'utilisation de cette instruction est possible mais elle ne présente aucun intérêt puisque AMSDOS détermine ces valeurs par lui-même. Même si vous indiquez donc un format déterminé, le format sera une nouvelle fois déterminé, avant l'accès à la disquette.

```
100         LD    HL,COMMAND        ;adresse de l'instruction
110         CALL  KL-FIND-COMMAND   ;vecteur &BCD4
120         RET   NC                ;pas trouvé instruction
```

```
130             LD     (FARADR),HL        ;ranger adresse de la routine
140             LD     A,C                ;sélection ROM dans l'accu
150             LD     (FARADR+2),A       ;et sauver également
160             LD     HL,FORMAT          ;marque du format (voir texte)
170             RST    &18
180             DEFW   FARADR             ;vecteur sur far adress 3 octets
190             RET
200  COMMAND:   DEFB   &83                ;nom de l'instruction
210  FARADR:    DEFS   3                  ;place pour l'adresse 3 octets
```

Ce programme ne se distingue en aucune manière de ceux qui ont été présentés jusqu'ici. Il reste simplement à définir comment les marques des différents formats doivent se présenter.
Cette distinction est entreprise au moyen des numéros de secteur. Avec le format CP/M-AMSDOS, il y a sur chaque piste d'une disquette 9 secteurs de 512 octets chacun. Ces 9 secteurs portent les numéros &41 à &49. Cela signifie que le bit 6 est mis sur chaque numéro de secteur. Dans le format de données AMSDOS par contre, les numéros de secteur sont &C1 à &C9. Dans ce cas, les bits 6 et 7 sont mis sur chaque numéro de secteur. En format IBM-CP/M, les numéros de secteur sont &01 à &08. En effet, avec ce format, seuls 8 secteurs sont formatés sur la disquette. Il y a donc ainsi un critère très simple pour distinguer entre les trois formats disponibles. Si vous transmettez à la routine &83 la valeur 0 (ou toute autre valeur dont les bits 6 et 7 soient annulés), c'est la table de paramètres pour le format IBM-CP/M qui sera alors mise en place dans la RAM. Le format CP/M-AMSDOS est mis en place lorsqu'une valeur avec un bit 6 mis et un bit 7 annulé est transmise. Cela pourrait donc être &40 mais aussi &73. Si vous mettez les deux bits supérieurs de la valeur à transmettre (par exemple &C0 ou &FF), c'est le format AMSDOS de données qui sera mis en place.

2.2.2.4 L'INSTRUCTION &84 READ SECTOR

La puissance de cette instruction ne doit pas être sous-estimée. Elle permet en effet de lire directement n'importe quel secteur de la disquette!
Vous n'êtes plus ainsi lié aux structures de fichier connues mais vous pouvez construire si nécessaire des structures qui vous soient entièrement propres comme par exemple des fichiers relatifs ou ISAM, puisque l'instruction suivante &85 permet en outre d'écrire directement sur n'importe quel secteur. Avec ces deux instructions, tous les octets

figurant sur la disquette sont à votre disposition en accès direct.

Pour pouvoir lire une secteur déterminé avec &84, certaines indications doivent être fournies à la routine.
Il s'agit tout d'abord du lecteur de disquette voulu. Cette indication doit être placée dans le registre E. Une valeur de 0 dans le registre E sélectionne le lecteur A, un 1 sélectionne le lecteur B.
Il faut d'autre part également indiquer à la routine le secteur voulu. C'est le registre C qui a été choisi comme registre pour ce paramètre. C'est toutefois le numéro de secteur figurant effectivement sur la disquette qui doit être transmis. Si vous voulez par conséquent lire le premier secteur d'une piste écrite en format CP/M-AMSDOS, le numéro de secteur correct sera donc &41. Le décalage sélectionné lors du formatage dans les bits 6 et 7 comme marque de format doit donc être inclu dans la valeur fournie.
Il faut enfin indiquer encore la piste (track) voulue sur la disquette. Le numéro de piste est indiqué dans le registre D. Dans l'AMSDOS les numéros de piste entre &00 et &27 (39 décimal) sont autorisés.

Voilà donc comment transmettre à travers les registres corrects tous les paramètres importants. Mais attention! Cela ne va pas tout à fait aussi vite. Nous devons encore décider dans quelle adresse mémoire doivent être écrits les 512 octets d'un secteur. Cette indication doit être transmise dans le registre HL. Nous avons ainsi réuni tous les paramètres et les données peuvent être envoyées.

```
100        LD    HL,COMMAND        ;adresse de l'instruction
110        CALL  KL-FIND-COMMAND   ;vecteur &BCD4
120        RET   NC                ;pas trouvé instruction
130        LD    (FARADR),HL       ;ranger adresse de la routine
140        LD    A,C               ;sélection ROM dans l'accu
150        LD    (FARADR+2),A      ;et sauver également
160        LD    E,DRIVE           ;0/1 pour drive A/B
170        LD    D,TRACK           ;0 à 39 pour piste voulue
180        LD    C,SECTOR          ;numéro de secteur avec décalage
                                   de format
190        LD    HL,BUFFER         ;512 octets pour les données du
                                   secteur
200        RST   &18               ;agit comme un CALL de la routine
                                   voulue
210        DEFW  FARADR            ;vecteur sur far adress 3 octets
220        RET
```

```
230 COMMAND:  DEFB  &84                  ;nom de l'instruction
240 FARADR:   DEFS  3                    ;place pour l'adresse 3 octets
```

Nous n'allons pas écrire maintenant un moniteur de disque complet. Le programme Basic suivant fournit cependant déjà la fonction 'lecture de secteurs'. Tapez donc ce programme, complétez-le et analysez-le pour vous familiariser avec la manipulation des instructions. Ce petit programme représente l'armature de base du moniteur de disque déjà évoqué.
Avant que vous n'essayiez de vous assurer du bon fonctionnement de ce programme, il faut absolument que vous fixiez la protection contre l'écriture sur la disquette que vous utiliserez. Il suffit en effet qu'un seul octet soit erronné, notamment l'octet de commande, pour que le secteur sélectionné ne soit pas lu mais remplacé par les données du buffer. Si vous remplacez par exemple de cette façon le premier secteur du catalogue de votre disquette master CP/M par un série de 0, il ne nous reste plus qu'à vous souhaiter que vous en ayez déjà fait une copie. Sinon 16 programmes de cette disquette seront sans aucun doute perdus. Suivez donc le conseil que nous vous donnons maintenant dans votre intérêt:

BACKUP, BACKUP, BACKUP !!!!

```
100 DEFINT a-z
110 MEMORY &A000-1
120 FOR adresse=&A000 TO &A01C
130 READ octet
140 POKE adresse,octet
150 controle=controle+octet
160 NEXT adresse
170 IF controle<>2941 THEN PRINT"erreur en datas!":END
180 MODE 2
190 INPUT"Canal de sortie (0/8)";periph
200 INPUT"Lecteur (0/1)";lecteur
210 INPUT"Piste (0-39)";piste
220 INPUT"Secteur (1-9)";secteur
230 POKE &A020,lecteur
240 POKE &A021,piste
250 POKE &A022,secteur+64
260 CALL &A000
270 MODE 2
280 lincnt=0
290 FOR i=&A030 TO &A030+511
300 IF lincnt=0 THEN PRINT#periph," ";HEX$(i-&A030,4);" ";
310 PRINT#periph,HEX$(PEEK(i),2);" ";:lincnt=lincnt+1
320 IF lincnt=16 THEN lincnt=0
330 a=PEEK(i):a=a AND 127
340 IF a<32 OR a=127 THEN t$="." ELSE t$=CHR$(a)
350 lin$=lin$+t$
360 IF lincnt=0 THEN PRINT#periph,lin$:lin$=""
370 NEXT
380 PRINT"Appuyez sur la barre <ESPACE> pour continuer"
390 IF INKEY(47) THEN 390
400 GOTO 180
410 DATA &21,&1c,&a0,&cd,&d4,&bc,&22,&1d
420 DATA &a0,&79,&32,&1f,&a0,&21,&20,&a0
430 DATA &5e,&23,&56,&23,&4e,&21,&30,&a0
440 DATA &df,&1d,&a0,&c0,&84
```

La manipulation du programme est expliquée par le programme lui-même. La transmission des paramètres se fait d'après le principe des boîtes à lettres, c'est-à-dire que nous écrivons les valeurs dans certaines cases mémoire d'où elles sont ensuite retirées par le programme machine. Cela est malheureusement nécessaire car il n'est pas possible de charger directement certaines valeurs dans les registres du Z80 à partir du Basic.
Après que les paramètres nécessaires aient été POKEs dans la mémoire, la routine en langage machine est appelée avec CALL. Le secteur indiqué est recherché sur la disquette et les données sont écrites dans un buffer de 512 octets qui commence en &A030. Les octets sont lus ici par le programme Basic qui les affiche sur l'écran en format hexadécimal. Vous pouvez également choisir de détourner la sortie sur une imprimante connectée à votre ordinateur.

2.2.2.5 L'INSTRUCTION &85   WRITE SECTOR

La simple lecture des données est déjà intéressante en soi. La routine que nous venons de vous présenter vous permet en effet un examen très intéressant du catalogue par exemple ou des pistes système 0 et 1 d'une disquette CP/M. Mais nous pouvons faire plus. L'écriture de données sur n'importe quel secteur de la disquette avec l'instruction &85 est aussi simple que la lecture avec l'instruction &84. Les paramètres nécessaires, le numéro du lecteur, la piste, le secteur et l'adresse du buffer de données sont même transmis à travers les mêmes registres que pour la lecture. Les modifications à apporter au programme précédent se ramènent donc à un seul octet. Seuls doivent être modifiés le numéro de

'instruction qui est le dernier élément de DATA dans le programme Basic ainsi bien sûr que la valeur de contrôle.

Vous pouvez expérimenter cette possibilité en prenant une disquette que vous venez de formater et en essayant d'écrire sur un secteur. Vous pouvez POKEr à cet effet n'importe quelles valeurs dans le buffer de secteur et écrire ensuite ce buffer sur un secteur. Avec la routine de lecture que nous vous avons fournie, vous pourrez ensuite contrôler si tout a bien marché comme vous le vouliez.

2.2.2.6 L'INSTRUCTION &86   FORMAT TRACK (FORMATER UNE PISTE)

Cette instruction est pour ceux d'entre vous qui sont des spécialistes. Elle permet de formater une piste isolée sur la disquette. Avec l'aide de l'instruction &86, vous pouvez intégrer dans vos programmes des routines de formatage, de sorte que l'utilisateur n'est pas obligé de se contenter du programme CP/M 'FORMAT.COM'.
Avant que nous n'expliquions plus en détail cette instruction 'cachée' de l'extension d'instructions, il nous faut expliquer comment le FDC 765 formate une piste. Pour ne pas déflorer la description à venir du FDC 765, nous ne vous révèlerons pour le moment que les points suivants:

Le controller du lecteur de disquette est, en ce qui concerne le formatage des disquettes, très axé sur les besoins des utilisateurs. Il se contente de certains octets, en petit nombre et il fait lui-même tout le 'sale boulot' comme par exemple la production des valeurs de contrôle, les différents marquages d'ID et ce qu'on appelle les GAPs. Il observe d'autre part quand l'orifice d'index marque le début d'une piste et il reconnaît par lui-même si la disquette est ou non protégée contre

l'écriture. Si ce composant rusé doit formater une piste, il a d'abord besoin de l'instruction correspondante. Cette instruction lui dit (entre autre) quelle piste il doit formater, sur quel lecteur, quelle est la taille des secteurs à produire et quelle valeur il doit utiliser comme octet de remplissage. L'octet de remplissage est une valeur qui figure, après un formatage réussi, sur tous les secteurs, en guise de données.
Il est évident que le moteur du lecteur doit être mis en marche pour un formatage. Dès donc qu'après l'instruction l'orifice d'index est reconnu, commence la procédure de formatage véritable. Le FDC attend du processeur 4 octets pour chaque secteur à formater. Pour chaque secteur doivent être indiqués les numéros de piste, de tête et de secteur ainsi que la taille du secteur. Du fait que le numéro de secteur doit être également indiqué pour chaque secteur, les secteurs peuvent être placés sur la disquette dans un ordre fixé par le programmeur. Cela peut considérablement accélérer les accès ultérieurs.

Mais nous en dirons plus à ce sujet plus tard. Nous allons maintenant voir ce que tout cela a à voir avec l'instruction &86.
L'instruction &86 attend des paramètres dans différents registres, comme les routines de lecture et d'écriture de secteur que nous avons décrites précédemment. Il y a d'abord la piste voulue. Cette valeur est transmise à travers le registre D. Dans le registre E doit se trouver la valeur du lecteur voulu. Le registre C reçoit le numéro du premier secteur à formater et le registre double HL est utilisé comme pointeur sur une table.
Les trois premiers registres et leurs fonctions ne se distinguent pas de ceux des routines décrites précédemment. Mais le registre double HL a lui aussi une fonction comparable à celle qu'il a pour l'écriture des données. En effet, dans la table adressée par HL figurent pour chaque secteur à formater 4 octets qui sont les valeurs déjà indiquées pour la piste, la tête et le lecteur, le numéro de secteur et la taille du secteur. L'exemple de programme suivant vous montre comment une piste isolée peut être formatée.

```
100                             ;FORMATAGE D'UNE PISTE ENTIERE
110  START:   LD    E,DRIVE     ;numéro de lecteur voulu dans le
                                registre E
120           LD    D,TRACK     ;numéro de piste dans le registre D
130           LD    C,&41       ;premier numéro de secteur sur la
                                piste
140           LD    HL,FTAB     ;table des 32 octets pour le FDC
150           RST   &18         ;Call Format &87
```

```
         DEFW  FORMAT      ;adresse de far adress de trois octets
         RET               ;la piste est formatée
FORMAT:  DEFW  &C652       ;adresse de la routine &87 dans l'AMSDOS
         DEFB  7           ;adresse nécessaire pour sélection ROM
FTAB:    EQU   $
SECT1:   DEFB  TRACK       ;numéro de piste pour ID de secteur
         DEFB  HEAD        ;numéro de tête du lecteur
         DEFB  &41         ;numéro du secteur
         DEFB  2           ;taille du secteur pour l'ID
SECT2:   DEFB  TRACK
         DEFB  HEAD
         DEFB  &43
         DEFB  2
SECT3:   DEFB  TRACK
         DEFB  HEAD
         DEFB  &45
         DEFB  2
SECT4:   DEFB  TRACK
         DEFB  HEAD
         DEFB  &47
         DEFB  2
SECT5:   DEFB  TRACK
         DEFB  HEAD
         DEFB  &49
         DEFB  2
SECT6:   DEFB  TRACK
         DEFB  HEAD
         DEFB  &42
         DEFB  2
SECT7:   DEFB  TRACK
         DEFB  HEAD
         DEFB  &44
         DEFB  2
SECT8:   DEFB  TRACK
         DEFB  HEAD
         DEFB  &46
         DEFB  2
SECT9:   DEFB  TRACK
         DEFB  HEAD
         DEFB  &48
         DEFB  2
```

2.2.2.7 L'INSTRUCTION &87 SEEK TRACK (RECHERCHE DE PISTE)

Bien que toutes les instructions avec accès direct à la disquette vues jusqu'ici se déplacent d'elles-mêmes vers la piste voulue, il peut être intéressant pour certaines applications de pouvoir positionner la tête sur une piste déterminée. Cette tâche est accomplie par l'instruction &87.
Pour le positionnement, deux paramètres seulement sont requis. Il faut d'abord indiquer le numéro du lecteur voulu. Le second paramètre exigé est le numéro de la piste vers laquelle la tête doit se déplacer.
Pour cette routine également, les programmeurs ont utilisé les registres comme interface. Le numéro de lecteur doit être transmis dans le registre E alors que le registre D doit contenir le numéro de piste.
L'utilisation des flags comme marque du succès de la routine fonctionne également comme à l'habitude. Un Carry mis vous indique que la piste recherchée sur la disquette a été trouvée.
Comme le programme nécessaire ne se distingue pas non plus fondamentalement des routines présentées précédemment, nous renoncerons ici pour une fois à le réécrire.

2.2.2.8 L'INSTRUCTION &88 TEST DRIVE

Voulez-vous savoir dans un programme machine si un lecteur de disquette déterminé est disponible? Rien de plus simple que cela. L'instruction &88 révèle toutes les informations importantes sur le lecteur sélectionné. Cette routine représente toutefois une exception dans une certaine mesure puisqu'elle n'attend pas l'indication du lecteur à tester comme d'habitude dans le registre E mais dans l'accumulateur.

Au retour de cette routine les flags et le contenu de l'accumulateur permettent de déterminer si le lecteur de disquette sélectionné est disponible. En cas d'exécution correcte de la routine, le flag Carry sera mis. L'accumulateur contient alors le contenu du registre d'état 0 du FDC. La seule chose intéressante à cet égard c'est que si le lecteur de disquette sélectionné est disponible l'accumulateur contiendra 0 pour le drive A ou 1 pour le drive B.

Vous trouverez une explication détaillée des autres valeurs, qui surviennent en cas d'erreur, dans le chapitre sur la programmation du FDC.

2.2.2.9 L'INSTRUCTION &89   RETRY COUNT

Si la tête doit être positionnée sur la disquette sur une piste déterminée, cela peut être réalisé très aisément avec l'instruction &87. Avec cette instruction on indique en effet au controller la piste voulue. Un registre spécial du FDC contient le numéro de piste actuel et le FDC détermine à partir de cette valeur et de la nouvelle piste indiquée le nombre d'impulsions nécessaires pour le stepper motor du lecteur de disquette. Après que le lecteur ait déplacé la tête de lecture/écriture du nombre de pas correspondant dans la direction voulue, le FDC lit automatiquement le numéro de piste qui a été placé sur la piste lors du formatage.
Si le numéro lu correspond au numéro voulu, l'instruction s'est achevée avec succès. Tout autre est le cas où aucune information ne peut être lue ou bien où le numéro de piste lu est différent du numéro voulu. Dans ce cas une nouvelle tentative pour trouver la piste voulue sera entreprise.
C'est exactement ici qu'intervient l'instruction &89. Cette instruction permet de fixer le nombre de tentatives de lecture après positionnement. L'AMSDOS prévoit une valeur par défaut de 10 tentatives. C'est normalement suffisant. Il peut toutefois arriver qu'il soit nécessaire d'augmenter ce nombre.
La nouvelle valeur voulue est transmise dans l'accumulateur à la routine &89. Il faut cependant noter à cet égard que la valeur 0 est comprise comme la valeur maximale 256. Il ne sert pas à grand chose d'employer des valeurs de 1 à 9 mais on peut par contre ressusciter parfois des disquettes présentant des problèmes de lecture en utilisant un nombre de tentatives de lecture supérieur à 10.

Pour expérimenter rapidement et sans assembleur les effets de cette routine, vous pouvez essayez de POKEr la valeur 0 dans la case mémoire &BE66. Cela revient à l'appel de l'instruction &89 avec transmission de 0 à travers l'accumulateur. Si vous placez maintenant une disquette brute, non formatée dans le lecteur, et que vous essayez de faire afficher le catalogue, vous entendrez très distinctement comment la tête "s'acharne" sur la disquette pour trouver la piste voulue. Si vous POKEz par contre un 1 dans la case mémoire &BE66, vous verrez que le lecteur de disquette se donne beaucoup moins de mal pour essayer de trouver des données sur cette disquette vide.

2.3 LES MESSAGES D'ERREUR DES ROUTINES DISK

Rien ni personne n'est parfait. L'AMSDOS doit donc lui aussi être prêt à faire face dans tous les endroits importants à des erreurs de formes tout à fait différentes. Lorsque des erreurs sont constatées, elles sont indiquées à l'utilisateur sous une forme ou une autre.
Pour autant que cela soit possible, les programmeurs ont essayé d'obtenir une très large compatibilité avec le travail sur cassette. Le meilleur exemple est à cet égard l'état des flags Zéro et Carry lors d'une exécution réussie. Comme pour les routines cassette correspondante, le flag Carry est en effet mis et le flag Zéro annulé dans ce cas. Les erreurs qui peuvent être détectées de même manière par les deux types de routines sont indiquées par le fait que les deux flags Carry et Zéro sont annulés.
Les erreurs qui ne peuvent être annoncées que par le lecteur de disquette sont indiquées par le fait que le flag Zéro est mis. En Basic 1.0 cet état du CPC 464 est interprété comme si la touche ESC avait été appuyée lors d'une opération cassette. Le programme en cours est donc interrompu et le message BREAK est sorti. En Basic 1.1, il est heureusement possible d'intercepter également ces erreurs avec ON ERROR GOTO, sans interruption du programme. Dans ce cas, la variable système DERR indique quelle erreur s'est produite. DERR ne correspond cependant PAS à la valeur de l'accumulateur après la routine disquette.
Nous allons maintenant étudier de plus près ce contenu de l'accumulateur car il est très important pour les programmeurs en assembleur. Les valeurs sont les mêmes pour le 464 et pour le 664.

&0E FILE NOT OPEN AS EXPECTED.
Ce message d'erreur peut être provoqué par des actions très différentes. Ce message survient en effet lorsque vous passez dans un fichier d'entrée d'une possibilité de lecture des données à l'autre. Vous obtenez également ce numéro d'erreur lors de la fermeture du fichier si vous avez appelé plus d'une fois la routine DISK OUT DIRECT.

&0F HARD END OF FILE.
Ce message d'erreur est sorti lorsque vous demandez d'autres données après la fin du fichier et que le buffer devrait être rempli avec de nouvelles données de la disquette.

&1A SOFT END OF FILE.
Ce message signale que vous avez atteint la fin du fichier. Vous devriez maintenant le fermer.

&20 BAD COMMAND.
Vous obtenez ce message par exemple lors d'entrées incorrectes de noms de fichier. Cela comprend par exemple les points d'interrogation dans le nom de fichier pour OPENIN. L'indication de lecteurs non autorisés ou de numéros user trop élevés provoque également ce message d'erreur.

&21 FILE ALREADY EXISTS.
Ce message d'erreur ne survient qu'en relation avec le changement de nom d'un fichier. Si vous inversez l'ordre des noms de fichier dans une instruction de changement de nom, AMSDOS sortira ce message d'erreur.

&22 FILE DOESN'T EXIST.
On obtient ce numéro d'erreur lorsqu'on tente d'accéder à des fichiers qui n'existent pas. La cause de l'erreur peut être que vous vous êtes trompé de disquette ou que le nom de fichier n'a pas été entré correctement.

&23 DIRECTORY FULL.
Une fois que 64 fichiers ont été entrés, le catalogue (directory) est plein. Même s'il y a encore des blocs libres sur la disquette, plus rien ne peut être sauvegardé sur la disquette.

&24 DISC FULL.
Tous les blocs de la disquette sont affectés à des fichiers. La sauvegarde de données supplémentaires a été interrompue.

&25 DISC HAS BEEN CHANGED WITH FILES OPEN.
Si vous obtenez ce numéro d'erreur lors de la lecture de données, cela n'est pas très grave. Les conséquences deviennent cependant fatales si vous changez de disquette alors qu'un fichier de SORTIE est encore ouvert. Ce fichier ne sera pas alors correctement fermé. Vous pouvez perdre jusqu'à 2047 octets qui n'ont pas été transférés du buffer sur la disquette.

&26 FILE IS READ ONLY.
Ce numéro d'erreur apparaît dans l'accumulateur lorsque vous avez tenté de modifier le nom d'un fichier protégé contre l'écriture ou lorsque vous avez tenté de supprimer un tel fichier.

Il convient d'ajouter après ces numéros d'erreur une remarque essentielle. Si l'erreur a déjà été communiquée, si donc elle a déjà été sortie sur l'écran, le bit 7 de son code d'erreur est mis. Le numéro

d'erreur &24, Disk full, deviendrait donc par exemple &A4.

Outre les erreurs 'logiques' de l'AMSDOS, il y a cependant également les erreurs 'physiques' du FDC. Ces erreurs concernent par exemple les erreurs de lecture, les erreurs d'écriture à cause de la protection contre l'écriture ou le cas où aucune disquette n'a été placée dans le lecteur. Ces messages d'erreur peuvent être reconnus par le fait que le bit 6 du numéro d'erreur est mis. Les erreurs possibles sont marquées par les bits du numéro d'erreur. La signification des différents bits est:

BIT 0 ADRESS MARK MISSING.
Cette erreur signale une disquette non formatée.

BIT 1 NOT WRITABLE.
La disquette est protégée contre l'écriture au moment d'un accès en écriture. Notez que cette erreur ne survient pas encore lors d'un OPENOUT car OPENOUT n'est pas encore écrit sur la disquette.

BIT 2 NO DATA.
Le secteur indiqué n'a pas pu être trouvé. Cette erreur ne se produira qu'en liaison avec des routines programmées par l'utilisateur pour la lecture et l'écriture directe de secteurs.

BIT 3 DRIVE NOT READY.
Il n'y a probablement pas de disquette dans le lecteur appelé. Une autre possibilité pouvant entraîner cette erreur est un accès à un lecteur non présent.

BIT 4 OVERRUN ERROR.
Cette erreur devrait être particulièrement rare car elle ne peut en fait pas survenir étant données les routines de secteur existantes. Si vous ^etes cependant gagné par l'ambition d'écrire vos propres routines de lecture/écriture directe de secteur, cette erreur peut alors par exemple survenir si votre routine est trop lente ou si vous négligez d'interdire les interruptions.

BIT 5 DATA ERROR.
Ce numéro d'erreur est le pire qui puisse survenir lors du travail sur disquette. Dans ce cas, en effet, le secteur indiqué n'est pas lisible.

Pour ces messages d'erreur également, le bit 7 signale si l'erreur a déjà été communiquée.

CHAPITRE 3: DONNEES TECHNIQUES DU LECTEUR DE DISQUETTE ET DE LA DISQUETTE

3.1 LE DISK CONTROLLER

Alors que sur le CPC 664 l'électronique de l'ordinateur et l'interface disquette sont intégrées dans le boîtier de l'ordinateur, sur le CPC 464, l'électronique du lecteur de disquette figure dans un petit boîtier enfiché dans le port d'extension. Nous allons décrire ici l'électronique de l'interface disquette externe dont vous trouverez le schéma dépliable en annexe de cet ouvrage. Les connexions sont sensiblement les mêmes pour les deux ordinateurs, de sorte que même les possesseurs d'un CPC 464 peuvent profiter de la description suivante. Nous évoquerons les principales différences lorsque nous traiterons des endroits correspondants. Pour le moment, le schéma de fonction 3.1.1 suffira à nous donner un aperçu des éléments de l'interface disquette.

Le point central du controller board est constitué par le floppy disk controller (FDC) PD 765. Ce circuit intégré constitue l'interface entre les lecteurs et le processeur du CPC. On peut certes tout à fait construire des lecteurs de disquette sans FDC mais la grande 'intelligence propre' du FDC simplifie grandement la construction. L'électronique nécessaire ainsi que l'importance du logiciel d'exploitation sont considérablement réduites par l'emploi d'un FDC. Un exemple éclairera ce point.
Le lecteur de disquette 1541 de la firme Commodore que beaucoup d'entre vous connaissent certainement comme lecteur de disquette du Commodore 64 est un lecteur de disquette construit sans FDC. Sans compter la lenteur de la transmission des données dûe à la construction même du lecteur (lenteur qui ne peut que faire sourir les possesseurs d'un CPC), l'investissement électronique pour ce lecteur est beaucoup plus important que sur le lecteur de disquette du CPC. L'électronique digitale du 1541 contient un processeur propre, des circuits intégrés périphériques de 40 pôles et une masse de circuits intégrés TTL de toute sorte. Cette masse de composants correspond à celle que requiert un CPC 664 complet!
Le logiciel d'exploitation du 1541 est, avec 16 K, deux fois plus grand que l'AMSDOS. Il est évident que les développeurs (pour des raisons de confort) et les commerçants (pour des raisons de coût) recourent volontiers aux FDCs dont l'utilisation est si pratique.
Avant que nous n'ôtions pour ainsi dire dans les pages suivantes le couvercle du FDC, nous allons jeter un regard sur le schéma et examiner les fonctions de base.

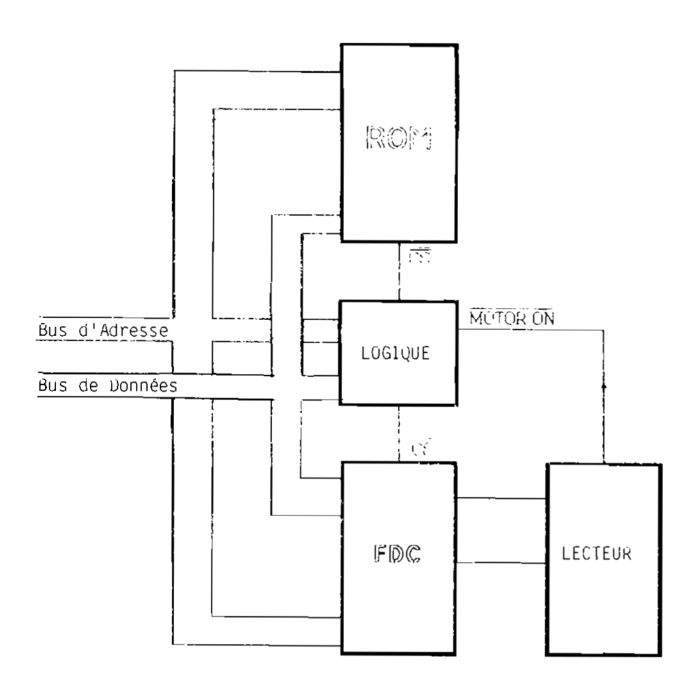

ROM
MOTOR ON
Bus d'Adresse
LOGIQUE
Bus de Données
FDC
LECTEUR

FIGURE 3.1.1

3.1.1 DESCRIPTION DE L'ELECTRONIQUE DU FLOPPY CONTROLLER BOARD

L'électronique complète du controller externe du 464 se compose de 12 circuits intégrés, 2 transistors, 20 résistances, 15 condensateurs et une diode, le nombre de composants étant comparable sur le 664. Ce petit nombre de composants n'a pu être obtenu que par la haute intégration de trois circuits intégrés. Il s'agit du FDC, du séparateur de données et de la ROM avec l'AMSDOS. Nous avons déjà parlé des deux premiers circuits intégrés. Intéressons-nous à la ROM AMSDOS. Les indications sur les noms des circuits intégrés correspondent à ceux du disk controller du 464. Il n'y a cependant sur ce plan pas de différences fonctionnelles importantes entre 464 et 664.

Bien que sur le CPC les ROMs externes ne puissent avoir qu'une taille de 16 K, le fait qu'un circuit intégré de 28 pôles ait été employé peut légitimement faire supposer que la ROM intégrée est une ROM de 32 K.
Nous ne savons pas à vrai dire s'il s'agit effectivement d'une ROM si généreusement employée ou s'il s'agit simplement d'un habitacle spécial. Toutefois l'emploi d'une ROM aussi grande comporte un avantage essentiel. Comme la ROM réside sur un socle de 28 pôles, il est possible de modifier le système d'exploitation et de le remplacer sans bricolage par une EPROM 16 K.
Et il y a dans les 16 K utilisés plus de place qu'il n'en faut pour vos propres extensions. L'AMSDOS n'occupe qu'à peine 8 K de la place mémoire disponible. Les 8 K restants sont nécessaires pour LOGO qui ne laisserait sinon presque plus de place pour les programmes LOGO sans l'emploi de cette zone de la ROM. Si l'on accepte cependant de renoncer au langage de programmation LOGO, un système d'exploitation étendu du lecteur de disquette peut donc avoir jusqu'à une taille de 16 K.

Pour adresser 16 K, il faut au total 14 canaux d'adresse. Ces 14 connexions de la ROM sont reliées directement à la fiche s'enfichant dans le port d'extension du CPC. Les 8 canaux de données sont également placés directement, sans circuits intégrés buffer, sur le bus de données du processeur. Le chip est activé par le signal ROMEN du port d'extension du CPC. Les sorties de données ne sont toutefois effectivement placées sur le bus de données que lorsque se produit un low (niveau faible) du signal OE (Output Enable: autoriser sortie). Cette distinction est particulièrement importante sur le CPC puisque le signal ROMEN devient également low lors d'accès du processeur au kernal et au Basic.
La production du signal OE est relativement simple du point de vue de l'électronique. Un instruction OUT sur l'adresse &DFxx avec une valeur de

7 suffit pour que la sortie Q (pin 5) du flip-flop IC 112-1 devienne high.
L'octet de donnée 7 (cela correspond à l'adresse de sélection de ROM pour ce qu'on appelle les far calls) entièrement décodé par les triples portes logiques Nor IC 111-1 et 111-2, la porte logique Nand IC 110-3 et les trois portes logiques XOR IC 109-1 à 109-3. Le décodage de l'adresse de port &DFxx est par contre effectué de façon très simple et incomplète. On teste simplement pendant l'instruction OUT si le bit d'adresse A13 est bien sur un niveau low.
Si ces deux conditions sont remplies, un high apparaît sur la sortie du flip-flop dont nous avons parlé. Toute autre valeur sur le bus de données provoque un low sur la sortie du flip-flop et ainsi la déconnexion de la ROM AMSDOS. La sortie du flip-flop est reliée aux deux portes logiques IC 110-1 et IC 106-1. L'autre entrée des deux portes logiques est reliée au canal d'adresse A15. Si maintenant le processeur veut accéder, alors que la sortie du flip-flop est mise, à une adresse de la ROM (ce qu'indique le signal ROMEN) située dans la zone de &C000 à &FFFF, la sortie de l'IC 110-1 devient low et la ROM AMSDOS obtient un niveau low sur la connexion OE. Simultanément, la sortie de la porte logique IC 106-1 devient high et la ROM Basic située à ces adresses est déconnectée à travers la diode D101.

Si vous suivez encore une fois sur le schéma les branchements de décodage de l'adresse de sélection de ROM, vous constatez que les trois portes logiques XOR de l'IC 109 sont placées ensemble, avec une entrée chacune, sur la masse à travers le pont LK 1. Que se passe-t-il donc lorsque ce pont n'est pas fermé?
Dans ce cas l'adresse de sélection de ROM ne sera plus 7 mais 0! S'il n'y avait que cela, ce ne serait pas sans grande conséquence. Beaucoup plus intéressant est le fait que l'AMSDOS teste l'adresse de sélection de ROM lors de l'initialisation, c'est-à-dire lors d'un Reset ou à la mise sous tension. Si une valeur de 0 est trouvée, on essaie automatiquement de charger et de lancer automatiquement CP/M à partir de la disquette. Si dans ce cas le programme CP/M 'AMSDOS.COM' n'est pas disponible, il n'est pas possible de travailler avec l'ordinateur sous le Basic Locomotive. Tout nouveau Reset ne fait que lancer à nouveau CP/M. Mais même avec le programme 'AMSDOS.COM', seul le Basic cassette peut être utilisé car les modifications de vecteurs nécessaires des routines CAS ne peuvent être exécutées dans ce cas.

Le second flip-flop que comporte l'IC 112 est également utilisé. C'est avec ce flip-flop que sont commandés les moteurs du lecteur de disquette.

L'entrée de données du flip-flop est placée sur le bit de donnée 0. L'impulsion CLK pour le flip-flop est obtenue à travers plusieurs portes logiques à partir des bits d'adresse A7, A8 et A10 ainsi que des signaux IORQ et WN. Les bits d'adresse indiqués doivent être low pour basculer le flip-flop. Cela donne une adresse de port de &FA7x, dans l'AMSDOS le flip-flop est appelé à travers le PORT &FA7E.

Les portes logiques pour décoder l'adresse de port du flip-flop du moteur sont également utilisées pour le décodage des adresses du FDC. Dans ce cas cependant, le bit d'adresse A8 doit être high pour générer le signal CS. Cela donne l'adresse de port &FB7X. Comme le bit d'adresse A0 du processeur est utilisé pour la sélection des deux registres du FDC, on obtient les adresses de port &FB7E pour le registre d'état principal et &FB7F pour le registre de données.

Un autre branchement intéressant du controller est celui construit avec les trois portes logiques IC 105-2, IC 110-4 et IC 109-4. Sur l'entrée pin 13 de la porte logique AND IC 105-2 se trouve la fréquence d'horloge du processeur fournie par le CPC, une fréquence de 4 MHz. Ce circuit intégré ne constitue qu'un buffer du signal d'horloge. Sur la sortie pin 11 de ce buffer est d'une part placée l'entrée pin 4 de la porte logique IC 109-4 mais une combinaison de deux résistances, d'un condensateur et de la porte logique NAND IC 110-4 est en outre encore branchée sur la sortie. La sortie de la porte logique NAND est branchée sur la deuxième entrée de la porte logique XOR, pin 5.
Si l'on examine le diagramme logique d'une porte logique XOR, on voit que la sortie d'une tel élément est toujours low si les deux entrées on le même niveau. La sortie devient par contre high lorsque les niveaux sont différents.
S'il n'y avait pas dans ce branchement la combinaison RC et la porte logique NAND, les deux entrées auraient toujours systématiquement le même niveau, la sortie du XOR resterait low. Mais comme le signal d'horloge est un peu ralenti par la combinaison RC et la porte logique NAND, il en résulte un effet intéressant de ce branchement. La fréquence d'entrée de 4 MHz est doublée. Sur la sortie du XOR, pin 6, nous obtenons une fréquence d'horloge de 8 MHz.
C'est avec cette fréquence qu'est commandée l'entrée d'horloge du séparateur de données qui produit par division de fréquence, à partir de ce signal, les deux fréquences d'horloge nécessitées par le FDC. Comme la connexion MINI (pin 3 du 9229) est placée sur la masse, la fréquence du signal CLK du FDC (pin 19) est de 4 MHz, le WCK (pin 21 du 765) est de 500 KHz.

Toutes les autres portes logiques employées dans le controller board font simplement office de buffer des signaux envoyés au lecteur de disquette. Il ne reste en fait d'intéressant que la production des deux signaux de sélection de lecteur de disquette. Bien que le FDC dispose de deux connexions séparées pour la production du Drive Select (sélection de lecteur), une seule connexion est utilisée. Le signal de sortie du pin 29, USO, est utilisé comme signal de sélection pour le lecteur B à travers une porte logique NAND branchée comme inverseur et comme signal de sélection pour le lecteur A à travers une autre porte logique NAND/inverseur. Suivant la polarité du signal USO, c'est donc le lecteur A ou le lecteur B qui est actif. Cette structure facilite vraiment les choses pour les programmeurs de l'AMSDOS. Il suffit donc de décrémenter la valeur de sélection de lecteur pour déterminer quel lecteur doit être activé. Si la valeur n'est pas nulle après décrémentation, c'est que c'était le lecteur A qui était actif, si la valeur est par contre nulle, c'est que c'est le lecteur B qui est actif.

3.1.2 LE FDC 765

Le FDC exploité par les firmes NEC sous le nom de PD 765, ROCKWELL sous le nom de R 6765 et INTEL sous le nom de 8765, peut être considéré comme un microprocesseur très spécialisé. Les possibilités de ce circuit intégré sont si étendues et si complexes que ce qualificatif n'est certainement pas exagéré.

Le format de données utilisé par le FDC correspond au format IBM 3740 en densité simple et au format IBM System 84 en double densité. De ce fait, les disquettes Commodore ou Apple par exemple ne peuvent malheureusement pas être lues ni écrites.
Avec ces 40 pins, il fournit tous les signaux nécessaires pour exploiter les lecteurs du marché des tailles 8", 5 1/4" et 3". Les signaux de commande disponibles permettent au développeur de connecter ce FDC à presque n'importe quel processeur. Deux possibilités fondamentales de connexion et d'exploitation sont offertes. La première méthode est l'exploitation DMA. En liaison avec un DMA controller, le FDC peut prendre en charge le contrôle de la mémoire du système informatique pour le transfert de données en lecture et en écriture. Il retire alors de la mémoire, à l'aide du DMA controller, les nouvelles données nécessitées ou écrit dans la mémoire, également en contournant le processeur, les données lues sur la disquette. Cette très rapide méthode de transfert de données n'est cependant pas utilisée sur le CPC et nous ne l'avons

évoquée que par souci d'exhaustivité.

Avec la seconde méthode, celle utilisée sur le CPC, le transfert de données est pris en charge par le processeur. Pour cette seconde méthode, il faut cependant à nouveau distinguer entre deux possibilités d'exploitation du FDC.
Il y a d'abord la méthode des interruptions. Pour chaque transfert de données, une interruption est alors produite. Dans la routine d'interruption du processeur doit alors être fourni ou lu par le processeur le prochain octet de donnée ou d'instruction. Du fait de la structure électronique du CPC, il ne pouvait non plus être question de cette méthode, de sorte que les développeurs ont choisi la méthode polling. Le processeur doit alors examiner régulièrement dans les registres du FDC quelle est la prochaine action demandée par le FDC.

Mais considérons tout d'abord un aperçu des données techniques du 765. Gardez cependant à l'esprit que les développeurs du controller board n'ont pas utilisé toutes les possibilités du 765.

* longueur de secteur programmable
* toutes les données du lecteur programmables
* jusqu'à quatre lecteurs connectables
* transfert de données au choix, en mode DMA ou pas en mode DMA
* connectable à presque tous les types de processeur courants
* alimentatlon électrique simple 5 volts
* horloge monophase simple de 4 ou 8 MHz
* habitacle de 40 pôles du circuit intégré

Nous allons maintenant nous intéresser un peu plus en détail au dernier point de cette brève présentation.

3.1.2.1 L'AFFECTATION DES CONNEXIONS DU FDC

Les connexions du FDC 765 peuvent être subdivisées en plusieurs groupes. Le premier groupe de connexions représente l'interface avec le processeur système. C'est donc à travers ces connexions que le FDC est commandé par le processeur.

Le deuxième groupe n'est nécessaire qu'en liaison avec l'exploitation DMA. C'est à travers ces signaux que communiquent le DMA controller et le FDC.

L'interface avec les lecteurs de disquette est constituée par le troisième groupe, qui est avec 19 connexions le groupe le plus important en nombre.

Dans le quatrième et dernier groupe peuvent être regroupées les connexions pour l'alimentation électrique et l'horloge.

Commençons l'examen des connexions par le premier groupe, l'interface avec le processeur.

L'interface avec le processeur

RESET : L'entrée RESET du FDC est active high. En exploitation normale, cette connexion est placée sur masse. Un high sur le pin RESET place le FDC dans un état déterminé.

CS* : CHIP SELECT. Un low sur ce pin sélectionne le FDC. Ce n'est qu'avec CS* = low que RD* et WR* deviennent valables pour le FDC. Comme la production du CS est à la libre disposition du développeur, le FDC peut être appelé au choix Memory-Mapped, donc comme élément de la zone de la mémoire, ou à travers des adresses de port.

RD* : READ*. Cette connexion doit être reliée au signal RD* du processeur. Dès que le processeur veut lire des données à partir du FDC, ce canal est mis sur low.

WR* : WRITE*. De même que le canal RD* signale des accès en lecture du processeur, un low sur WR* indique que le processeur écrit des données ou des instructions dans le FDC.

AO : ADRESS LINE O. Le FDC ne dispose que de deux adresses pouvant être appelées de l'extérieur. La distinction entre les deux adresses est effectuée avec le signal AO. Ce canal est normalement relié au bit d'adresse le plus bas du processeur.

µPD 765A

RESET	VCC (+5V)
RD	RW/SEEK
WR	LCT/DIR
CS	FLTR/STEP
A0	HOLD
DB 0	READY
DB 1	WPRT/2 SIDE
DB 2	FLT/TRK0
DB 3	PS 0
DB 4	PS 1
DB 5	WDATA
DB 6	US 0
DB 7	US 1
DRQ	SIDE
DACK	MFM
TC	WE
INDEX	SYNC
INT	RDATA
ϕ	WINDOW
GND	WCLK

FIGURE 3.1.2.1.1

DB0 - DB7 : DATABUS 0-7. Ces connexions du FDC sont reliées au bus de données du système. Toutes les instructions et données sont transportées à travers ces huit connexions bi-directionnelles. La direction des données est chaque fois déterminée soit par le processeur, soit par le DMA controller en mode DMA.

INT : INTERRUPT. A travers cette connexion, le FDC peut produire une interruption du processeur du système. Les interruptions sont produites à chaque transfert d'octet (non connecté sur le CPC)

Signaux pour le mode DMA (inutilisé sur le CPC)

DRQ : DMA REQUEST. A travers cette connexion, le FDC signale au DMA controller qu'un accès à la mémoire doit se produire. A la prochaine occasion possible, le DMA controller prendra en charge le bus système. Le processeur est alors déconnecté.

DACK* : DMA ACKNOWLEDGE. Ce signal indique au FDC que le DMA controller a pris en charge le bus et a maintenant commencé le transfert de données.

TC : TERMINAL COUNT. Un niveau high sur cette connexion interrompt le transfert de données vers et à partir du FDC. Bien que cette connexion soit essentiellement utilisée en mode DMA, le transfert de données peut également être interrompu à travers cette connexion dans les systèmes commandés par interruption.

L'interface disquette

US0, US1 : UNIT SELECT 0/1. A travers ces deux connexions peuvent être connectés directement deux lecteurs de disquette, mais avec l'aide d'un décodeur deux-à-quatre, ce sont même quatre lecteurs qui peuvent être connectés. C'est à travers ces connexions qu'est appelé chaque fois le lecteur voulu pour l'écriture ou la lecture de données.

HD : HEAD SELECT. Comme le FDC est conçu pour l'exploitation de lecteurs de disquette à double tête de lecture, la sélection de la tête peut s'effectuer à travers cette connexion lorsqu'on utilise de tels lecteurs.

HDL : HEAD LOAD. Ce signal est employé presque exclusivement sur les lecteurs 8". Les moteurs de ces lecteurs ne sont pas mis en marche quand c'est nécessaire, mais ils tournent normalement sans arrêt. Mais pour ménager malgré tout la disquette et la tête d'écriture, la tête n'est normalement 'chargée', à travers un aimant qui l'amène près de la surface de la disquette, que lorsque c'est nécessaire. La commande de l'aimant s'effectue alors au moyen de HDL.

IDX : INDEX. Le signal produit par le faisceau lumineux est placé sur cette connexion. Il signale au FDC le début physique d'une piste.

RDY : READY. Le signal READY fourni par le lecteur de disquette indique qu'une disquette se trouve dans le lecteur de disquette et que celle-ci tourne à une vitesse minimum déterminée. Ce n'est qu'après apparition du READY que le FDC accède au lecteur de disquette.

WE : WRITE ENABLE. Cette sortie du FDC doit être high pour que des données puissent être écrites sur la disquette.

RW/SEEK : READ WRITE/SEEK. Un lecteur de disquette produit au total plus de signaux qu'il n'y en a de disponibles pour l'interface disquette sur un socle de 40 pôles. Toutefois, tous les signaux ne sont pas nécessaires en même temps à tout moment. Huit de ces signaux disquette ont été pour cette raison séparés en deux groupes qui peuvent être placés de façon sélective sur quatre connexions du FDC. Le FDC sélectionne de lui-même à travers la connexion RW/SEEK les signaux dont il a besoin à un moment donné.

FR/STP : FIT RESET/STEP. C'est le premier des quatre doubles signaux du FDC. Cette sortie a différentes significations suivant l'opération exécutée. D'une part cette connexion permet de restaurer le flip-flop d'erreur qui existe sur certains lecteurs. La seconde utilisation, beaucoup plus courante est la commande de l'entrée des pas du lecteur. A chaque déplacement de tête, les impulsions nécessaires sont fournies sur cette connexion.

FLT/TRO : FAULT/TRACKO. Cette entrée peut elle aussi évaluer deux

signaux différents. Si une opération SEEK (voir Programmation du FDC) est exécutée, le signal track0 du lecteur est attendu sur cette connexion. Ce signal est produit par un faisceau lumineux ou par un commutateur mécanique, lorsque la tête de lecture/écriture se trouve sur la piste physique 0. Le seconde fonction, le signal Fault est générée par certains lecteurs en cas d'erreur. Elle peut être à nouveau annulée par le FDC avec le signal FR/STP défini plus haut. Ce signal est contrôlé lors d'opérations Read/Write du FDC.

LCT/DIR : LOW CURRENT/DIRECTION. Les impulsions de pas de FR/STP indiquent bien sûr uniquement que la tête doit être déplacée. LCT/DIR détermine alors en mode Seek la direction du déplacement de la tête. La fonction LOW CURRENT est nécessaire lors de l'écriture des données. Ce signal permet de diminuer le flot d'écriture sur les pistes intérieures. Vous trouverez des détails sur ce signal dans la description des bases théoriques de la sauvegarde sur disquette.

WP/TS : WRITE PROTECT/TWO SIDE. Indépendamment des diverses méthodes utilisées avec les différentes tailles de lecteurs de disquette, l'état de protection contre l'écriture est communiqué par le lecteur de disquette au controller, sous forme d'un signal. Ce signal est testé par l'entrée WP/TS lors des opérations de lecture/écriture. Le signal TS est testé lors des opérations Seek. Il n'est nécessaire qu'en liaison avec les lecteurs à double tête de lecture.

WDA : WRITE DATA. Les données sérielles à écrire sont transmises à travers cette connexion au lecteur de disquette. Ce peuvent être aussi bien les données pour l'écriture d'un secteur que toutes les informations nécessaires lors du formatage.

PS0, 1 : PRE SHIFT 0/1. A travers ces connexions, le FDC indique pour le format à double densité (MFM) à une électronique appropriée comment le flot sériel de données doit être écrit sur la disquette. Les trois états possibles sont EARLY, NORMAL et LATE pour la précompensation.

RD : READ DATA. Les informations lues sur la disquette sont entrées dans le FDC à travers cette entrée. C'est à partir de ce flot sériel de bits que les octets écrits à l'origine sont reconstitués.

RDW : READ DATA WINDOW. Ce signal est obtenu dans un séparateur de données à partir des données lues. Plus de détails dans le chapitre Bases de la sauvegarde sur disquette.

VCO : VCO SYNC. Ce signal est nécessaire pour la commande du VCO dans le séparateur de données PLL.

MFM : MFM MODE. Cette connexion signale si le controller travaille en format simple densité (MF) ou double densité (MFM).

Alimentation électrique et signaux d'horloge

Vcc : +5 Volts. C'est à travers cette connexion que le FDC reçoit son alimentation en courant électrique. La tension de 5 Volts doit être constamment dans la zone de +-5 %. L'intensité de courant nécessaire est au maximum de 150 mA.

GND : GROUND. Connexion à la masse du FDC.

CLK : CLOCK. Le FDC a besoin d'une fréquence d'horloge. Suivant les lecteurs, cette fréquence doit être de 4 MHz (pour les 5 1/4 et les formats plus petits) ou de 8 MHz (pour les 8").

WCK : WRITE CLOCK. La fréquence de ce signal doit être sélectionnée en fonction du format de données choisi. Pour MF, la fréquence doit être de 500 kHz, pour MFM, de 1 MHz. Cette fréquence détermine la vitesse de transmission des données vers et à partir du lecteur de disquette.

3.1.2.2 LA PROGRAMMATION DU FDC 765

Le FDC 765 ne dispose vers l'extérieur que de deux adresses ou registres. Le niveau du signal A0 détermine quel registre est disponible. Si A0 est sur la masse, on peut accéder au registre principal d'état. Ce registre principal d'état ne peut être que lu. Un niveau haut sur A0 autorise l'accès au registre de données. Le registre de données peut être écrit ou lu. C'est à travers ce registre que le FDC est programmé, que toutes les données de lecture/écriture sont transférées et que le données sont fournies au processeur dans la phase résultat.

Il faut distinguer pour les instructions du FDC trois phases fondamentales. La première phase est la phase instruction. Dans cette phase, tous les paramètres nécessaires pour l'instruction sont transmis au FDC. Pour certaines instructions, cela peut représenter jusqu'à 9 octets. Dès que tous les octets de cette phase ont été communiqués au FDC, commence la phase exécution. Cela signifie concrètement que par exemple, après une instruction de lecture, les données arrivent maintenant de la disquette. Après que la phase exécution soit terminée, commence la dernière phase, la phase résultat. Pendant la phase résultat, le FDC fournit jusqu'à 7 informations d'état qui doivent toutes être lues par le processeur.
Le schéma indiqué ici ne vaut toutefois pas pour toutes les instructions. Pour certaines instructions, il n'y a pas de phase résultat alors que d'autres instructions n'ont pas de phase exécution. Il y a même une instruction qui ne dispose d'aucune de ces deux phases et qui se contente de la phase instruction.

Avant que nous n'en venions à traiter les différentes instructions, nous allons d'abord jeter un regard sur les informations d'état.
Le FDC 765 dispose au total de 7 registres d'état. Le registre principal d'état occupe une adresse propre (A0=low) comme nous l'avons déjà indiqué et il ne peut qu'être lu. Un accès à ce registre est toutefois possible à tout moment, même lors du traitement de l'instruction.
Les quatre autres registres d'état ne sont accessibles que dans la phase résultat de certaines instructions. Le premier octet dans la phase résultat de ces instructions indique l'état du registre d'état 0 (ST0). Les états de ST1 et ST2 sont fournis ensuite comme octets deux et trois dans la phase résultat. L'état de ST3, le dernier registre d'état, ne peut être obtenu que sur demande particulière, avec une instruction spéciale dont la seule fonction est de fournir l'état de ST3. Nous étudierons un peu plus tard la signification des registres d'état.

Le FDC 765 dispose au total de 15 instructions différentes. Ce simple nombre montre déjà clairement que ce controller maîtrise plus de choses que la simple lecture/écriture. Nous allons maintenant examiner en détail quelles possibilités il offre. Si certaines notions non expliquées dans ce chapitre, telles que par exemple ID de secteur, Gap ou Data Adress Mark ne vous disent rien, vous pouvez vous reporter pour ces notions au chapitre sur l'écriture physique des données.

LIRE LES DONNEES

Avant que le FDC puisse lire des données à partir de la disquette, 9 octets doivent lui être transmis dans la phase instruction. Après indication de toutes les données, le signal Head Load est activé et on attend le Head Load Time programmé. Le FDC lit ensuite les ID de secteur jusqu'à ce qu'il rencontre l'ID du secteur indiqué ou jusqu'à ce que l'impulsion de l'index apparaisse pour la deuxième fois depuis le début de la recherche. Dans le premier cas il commence la phase exécution, dans le second cas il met fin à l'instruction et la phase résultat commence.

Dès qu'il a identifié le secteur, il commence la phase exécution. Dans la phase exécution, les données sont lues sur la disquette et fournies au processeur à travers le bus de données. Les délais correspondants sont très courts. Environ toutes les 26 microsecondes un octet est disponible et doit être lu par le processeur.

Après transmission du dernier octet du secteur voulu, une impulsion TERMINAL COUNT (TC, pin 16) doit être placée sur le FDC pour déclencher la phase résultat. En effet, en l'absence d'une impulsion TC, ce qu'on appelle un Multi Sector Read sera exécuté.
Multi Sector Read signifie que le controller lit des données jusqu'à ce qu'il ait lu le dernier secteur de la piste. On peut par ailleurs renoncer à l'impulsion TC sous certaines conditions. Dans la phase instruction de 'lire secteur' et de quelques autres instructions, ce n'est pas seulement le secteur voulu qui doit être indiqué mais aussi le dernier secteur de la piste. Si alors les deux valeurs sont identiques, le FDC interrompt automatiquement la lecture à la fin du secteur voulu et commence la phase résultat.

PHASE	R/W	BUS DE DONNEES D7	D6	D5	D4	D3	D2	D1	D0	REMARQUES
		LIRE LES DONNEES								
INSTRUCTION	W	MT	MF	SK	0	0	1	1	0	Codes d'instruction
	W	X	X	X	X	X	HD	US1	US0	
	W	-----Numéro de piste-----------								Information ID secteur
	W	-----Adresse de tête-----------								avant exécution de
	W	-----Adresse de secteur--------								l'instruction
	W	-----Taille du secteur---------								
	W	--Dernier No secteur sur piste-								
	W	---Vide entre ID et données----								
	W	Longueur secteur si taille secteur = 0								
EXECUTION										Transfert de données entre FDD et le système
RESULTAT	R	----------Etat 0---------------								Information d'état
	R	----------Etat 1---------------								après exécution de
	R	----------Etat 2---------------								l'instruction
	R	-----Numéro de piste-----------								Information ID secteur
	R	-----Adresse de tête-----------								après exécution de
	R	-----Numéro de secteur---------								l'instruction
	R	-----Taille du secteur---------								

MT Bit Multitrack: lorsqu'il est mis, la fonction est poursuivie sur la seconde face de la disquette pour les fonctions multi-secteur; disponible uniquement sur les lecteurs à double tête, dans l'AMSDOS toujours 0.

MF Bit Mode MFM: lorsqu'il est mis, le FDC travaille en double densité, dans l'AMSDOS toujours 1.

SK Bit SKip: lorsqu'il est mis, les secteurs effacés (deleted DAM) sont sautés. Inutilisable sous AMSDOS et CP/M. Toujours 0.

HD Bit HeaD select: avec les lecteurs à double tête, la sélection de la face est faite à travers ce bit. Dans l'AMSDOS, toujours 0.

US 0,1 Unit Select: les lecteurs sont sélectionnés à travers ces bits. Sous AMSDOS, 0 pour lecteur A, 1 pour lecteur B.

R Read = lire

W Write = écrire

Les noms et abréviations donnés ici valent également pour toutes les présentations d'instruction suivantes.

Dans la phase résultat, 7 octets sont fournis par le FDC au processeur. Le processeur doit retirer, c'est-à-dire lire ces octets le plus vite possible. Le FDC n'accepte aucune nouvelle instruction avant que le dernier octet de la phase résultat n'ait été lu.

Les trois premiers octets fournis sont les états des registre d'état 0 à 2. Avec ces trois registres, le programmeur dispose de toutes les données nécessaires sur le succès ou l'échec de l'instruction.
Ensuite sont fournis le numéro de piste actuel, l'adresse de tête (importante pour les lecteurs à double tête), le numéro de secteur et la taille du secteur. Ce n'est qu'après cela qu'une nouvelle instruction peut être exécutée.

ECRIRE DONNEES
Comme pour la lecture, lors de l'écriture de secteurs, 9 octets sont nécessaires. Après que tous les octets aient été transmis au FDC, celui-ci commence (après écoulement du Head Load Time) la recherche du secteur voulu. Il lit à cet effet les IDs de secteur jusqu'à ce qu'il ait identifié le secteur voulu ou jusqu'à ce que l'impulsion d'index ne survienne pour la deuxième fois depuis le début de la recherche. Dans ce dernier cas, l'instruction est interrompue et la phase résultat commence immédiatement.
Si cependant le secteur voulu est rencontré, le FDC demande maintenant au processeur les données à écrire (pas en exploitation DMA). Après que toutes les données aient été écrites, la phase résultat commence. Elle ne

se distingue pas de celle de l'instruction Lire données.

PHASE	R/W	D7	D6	D5	D4	D3	D2	D1	D0	REMARQUES
		BUS DE DONNEES								
		ECRIRE DONNEES								
INSTRUCTION	W	MT	MF	0	0	0	1	0	1	Codes d'instruction
	W	X	X	X	X	X	HD	US1	US0	
	W	-----Numéro de piste-----------								Information ID secteur
	W	-----Adresse de tête-----------								avant exécution de
	W	-----Adresse de secteur--------								l'instruction
	W	-----Taille du secteur---------								
	W	--Dernier No secteur sur piste-								
	W	---Vide entre ID et données----								
	W	Longueur secteur si taille secteur = 0								
EXECUTION										Transfert de données entre FDD et le système
RESULTAT	R	----------Etat 0---------------								Information d'état
	R	----------Etat 1---------------								après exécution de
	R	----------Etat 2---------------								l'instruction
	R	-----Numéro de piste-----------								Information ID secteur
	R	-----Adresse de tête-----------								après exécution de
	R	-----Numéro de secteur---------								l'instruction
	R	-----Taille du secteur---------								

LIRE DONNEES EFFACEES

Le nom de cette instruction est curieux. Il ne faut pas comprendre ici par données supprimées les données d'un fichier supprimé. Quelle que soit l'intelligence du FDC, celui-ci n'a aucune idée de ce que sont les fichiers et les catalogues. Le terme 'effacées' se rapporte bien plutôt à la possibilité de marquer des secteurs comme effacés en entrant une Data Adress Mark. De tels secteurs sont alors ignorés en lecture et écriture

normales. La lecture de données effacées s'effectue pour le reste comme la lecture de données normales.

PHASE	R/W	BUS DE DONNEES D7	D6	D5	D4	D3	D2	D1	D0	REMARQUES
		LIRE DONNEES EFFACEES								
INSTRUCTION	W	MT	MF	SK	0	1	1	0	0	Codes d'instruction
	W	X	X	X	X	X	HD	US1	US0	
	W	-----Numéro de piste-----------								Information ID secteur
	W	-----Adresse de tête-----------								avant exécution de
	W	-----Adresse de secteur--------								l'instruction
	W	-----Taille du secteur---------								
	W	--Dernier No secteur sur piste-								
	W	---Vide entre ID et données----								
	W	Longueur secteur si taille secteur = 0								
EXECUTION										Transfert de données entre FDD et le système
RESULTAT	R	----------Etat 0---------------								Information d'état
	R	----------Etat 1---------------								après exécution de
	R	----------Etat 2---------------								l'instruction
	R	-----Numéro de piste-----------								Information ID secteur
	R	-----Adresse de tête-----------								après exécution de
	R	-----Numéro de secteur---------								l'instruction
	R	-----Taille du secteur---------								

ECRIRE DONNEES EFFACEES

Cette instruction ne diffère pas fondamentalement de l'écriture normale de données. La seule exception est le traitement spécial de la Data Adress Mark. Ici en effet, une Data Adress Mark effacée est entrée.

PHASE	R/W	BUS DE DONNEES D7	D6	D5	D4	D3	D2	D1	D0	REMARQUES
		ECRIRE DONNEES EFFACEES								
INSTRUCTION	W	MT	MF	0	0	1	0	0	1	Codes d'instruction
	W	X	X	X	X	X	HD	US1	US0	
	W	-----Numéro de piste-----------								Information ID secteur
	W	-----Adresse de tête-----------								avant exécution de
	W	-----Adresse de secteur--------								l'instruction
	W	-----Taille du secteur---------								
	W	--Dernier No secteur sur piste-								
	W	---Vide entre ID et données----								
	W	Longueur secteur si taille secteur = 0								
EXECUTION										Transfert de données entre FDD et le système
RESULTAT	R	----------Etat 0---------------								Information d'état
	R	----------Etat 1---------------								après exécution de
	R	----------Etat 2---------------								l'instruction
	R	-----Numéro de piste-----------								Information ID secteur
	R	-----Adresse de tête-----------								après exécution de
	R	-----Numéro de secteur---------								l'instruction
	R	-----Taille du secteur---------								

LECTURE D'UNE PISTE

Cette instruction est comparable à la lecture d'un secteur. Toutefois avec cette instruction, tous les octets de données de la piste sont lus. Cette instruction se termine soit si le secteur indiqué comme dernier secteur a été lu, soit si un tel secteur n'a pas été rencontré, si l'orifice d'index produit une seconde impulsion après le début de l'instruction.

PHASE	R/W	D7	D6	D5	D4	D3	D2	D1	D0	REMARQUES
		BUS DE DONNEES								
		LIRE UNE PISTE								
INSTRUCTION	W	0	MF	SK	0	0	0	1	0	Codes d'instruction
	W	X	X	X	X	X	HD	US1	US0	
	W	-----Numéro de piste-----------								Information ID secteur avant exécution de l'instruction
	W	-----Adresse de tête-----------								
	W	-----Adresse de secteur--------								
	W	-----Taille du secteur---------								
	W	--Dernier No secteur sur piste-								
	W	---Vide entre ID et données----								
	W	Longueur secteur si taille secteur = 0								
EXECUTION										Transfert de données entre FDD et le système. Le FDC lit tout ce qui se trouve sur la piste entre l'orifice d'index et EOT
RESULTAT	R	----------Etat 0---------------								Information d'état après exécution de l'instruction
	R	----------Etat 1---------------								
	R	----------Etat 2---------------								
	R	-----Numéro de piste-----------								Information ID secteur après exécution de l'instruction
	R	-----Adresse de tête-----------								
	R	-----Numéro de secteur---------								
	R	-----Taille du secteur---------								

FORMATAGE D'UNE PISTE

Le formatage d'une piste est très simple avec le 765. Une chaîne de 6 octets est à cet effet transmise au FDC dans la phase instruction. Lorsque, après la transmission complète des 6 octets, le FDC reconnaît à l'impulsion d'index le début physique de la piste, il commence automatiquement à formater la piste avec tous les Adress Marks, Gaps et IDs nécessaires.

L'octet numéro 4 de la chaîne d'instruction indique combien de secteurs doivent être disposés sur la piste. Pour chaque secteur, le FDC demande quatre autres octets. Un de ces octets est le numéro de secteur qui est inscrit dans l'ID secteur. Il est ainsi possible de formater les secteurs dans des ordres différents. Cela permet, par un choix judicieux de l'ordre, d'accélérer nettement les accès ultérieurs en lecture.

PHASE	R/W	D7	D6	D5	D4	D3	D2	D1	D0	REMARQUES
		BUS DE DONNEES								
		FORMATER UNE PISTE								
INSTRUCTION	W	0	MF	0	0	1	1	0	1	Codes d'instruction
	W	X	X	X	X	X	HD	US1	US0	
	W	-----Taille du secteur---------								Octets par secteur
	W	-----Secteurs par piste--------								Secteurs par piste
	W	---Vide entre ID et données----								Vide #3
	W	--Modèle données pour secteur--								Octet de remplissage
EXECUTION										Le FDC formate une piste entière.
RESULTAT	R	----------Etat 0---------------								Information d'état
	R	----------Etat 1---------------								après exécution de
	R	----------Etat 2---------------								l'instruction
	R	-----Numéro de piste-----------								Dans ce cas,
	R	-----Adresse de tête-----------								l'information ID n'a
	R	-----Numéro de secteur---------								pas de signification
	R	-----Taille du secteur---------								

LIRE ID

Cette instruction permet, après avoir fourni deux octets dans la phase instruction, de lire sur la disquette la prochaine ID possible. Chaque secteur a reçu sa propre ID lors du formatage. Cette ID contient les valeurs de numéro de secteur, de piste, de face de disquette et de taille de -secteurT.a iCb$ e

résultat finale en même temps que les états des registres d'état ST0 à ST2.

PHASE	R/W	D7	D6	D5	D4	D3	D2	D1	D0	REMARQUES
		BUS DE DONNEES								
		LIRE ID								
INSTRUCTION	W	0	MF	0	0	1	0	1	0	Codes d'instruction
	W	X	X	X	X	X	HD	US1	US0	
EXECUTION										Sauvegarder la première information ID correcte dans les registres de données.
RESULTAT	R	----------Etat 0---------------								Information d'état
	R	----------Etat 1---------------								après exécution de
	R	----------Etat 2---------------								l'instruction
	R	-----Numéro de piste-----------								Dans ce cas,
	R	-----Adresse de tête-----------								l'information ID n'a
	R	-----Numéro de secteur---------								pas de signification
	R	-----Taille du secteur---------								

LES INSTRUCTIONS SCAN, TEST D'UN SECTEUR

Ces instructions reviennent par leurs effets à un verify, c'est-à-dire à une vérification de l'identité entre les données écrites et les données à écrire. Les conditions de test possibles sont 'égal', 'supérieur ou égal' et 'inférieur ou égal'. Après que 9 octets aient été transmis dans la phase instruction, des données sont lues par le FDC sur le secteur sélectionné. En même temps, le FDC demande des données au processeur. Chaque octet de la disquette est comparé à un octet venant du processeur, suivant les conditions de test indiquées. Cette instruction se termine soit lorsque la condition de test est remplie dans le secteur indiqué, soit lorsque le dernier secteur de la piste a été testé ou lorsqu'une impulsion TC a été placée sur le pin 16.

PHASE	R/W	BUS DE DONNEES D7	D6	D5	D4	D3	D2	D1	D0	REMARQUES
		SCAN EQUAL								
INSTRUCTION	W	MT	MF	SK	1	0	0	0	1	Codes d'instruction
	W	X	X	X	X	X	HD	US1	US0	
	W	-----Numéro de piste-----------								Information ID secteur
	W	-----Adresse de tête-----------								avant exécution de
	W	-----Adresse de secteur--------								l'instruction
	W	-----Taille du secteur---------								
	W	--Dernier No secteur sur piste-								
	W	---Vide entre ID et données----								
	W	Longueur secteur si taille secteur = 0								
EXECUTION										Comparaison de données entre FDD et système
RESULTAT	R	----------Etat 0---------------								Information d'état
	R	----------Etat 1---------------								après exécution de
	R	----------Etat 2---------------								l'instruction
	R	-----Numéro de piste-----------								Information ID secteur
	R	-----Adresse de tête-----------								après exécution de
	R	-----Numéro de secteur---------								l'instruction
	R	-----Taille du secteur---------								

PHASE	R/W	D7	D6	D5	D4	D3	D2	D1	D0	REMARQUES
		BUS DE DONNEES								
		SCAN LOW OR EQUAL								
INSTRUCTION	W	MT	MF	SK	1	1	0	0	1	Codes d'instruction
	W	X	X	X	X	X	HD	US1	US0	
	W	-----Numéro de piste-----------								Information ID secteur
	W	-----Adresse de tête-----------								avant exécution de
	W	-----Adresse de secteur--------								l'instruction
	W	-----Taille du secteur---------								
	W	--Dernier No secteur sur piste-								
	W	---Vide entre ID et données----								
	W	Longueur secteur si taille secteur = 0								
EXECUTION										Comparaison de données entre FDD et le système
RESULTAT	R	----------Etat 0---------------								Information d'état
	R	----------Etat 1---------------								après exécution de
	R	----------Etat 2---------------								l'instruction
	R	-----Numéro de piste-----------								Information ID secteur
	R	-----Adresse de tête-----------								après exécution de
	R	-----Numéro de secteur---------								l'instruction
	R	-----Taille du secteur---------								

PHASE	R/W	D7	D6	D5	D4	D3	D2	D1	D0	REMARQUES
		BUS DE DONNEES								
		SCAN HIGH OR EQUAL								
INSTRUCTION	W	MT	MF	SK	1	1	1	0	1	Codes d'instruction
	W	X	X	X	X	X	HD	US1	US0	
	W	-----Numéro de piste-----------								Information ID secteur
	W	-----Adresse de tête-----------								avant exécution de
	W	-----Adresse de secteur--------								l'instruction
	W	-----Taille du secteur---------								
	W	--Dernier No secteur sur piste-								
	W	---Vide entre ID et données----								
	W	Longueur secteur si taille secteur = 0								
EXECUTION										Comparaison de données entre FDD et le système
RESULTAT	R	----------Etat 0---------------								Information d'état
	R	----------Etat 1---------------								après exécution de
	R	----------Etat 2---------------								l'instruction
	R	-----Numéro de piste-----------								Information ID secteur
	R	-----Adresse de tête-----------								après exécution de
	R	-----Numéro de secteur---------								l'instruction
	R	-----Taille du secteur---------								

RECALIBRER, CHERCHER PISTE ZERO

Avec cette instruction, la tête du lecteur sélectionné est déplacée vers la piste zéro, soit jusqu'à ce que l'impulsion Track0 du lecteur signale au FDC que cette piste a été atteinte, soit jusqu'à ce que le FDC ait fourni 77 impulsions de pas. Les registres d'état permettent de déterminer comment cette instruction s'est achevée.

PHASE	R/W	BUS DE DONNEES								REMARQUES
		D7	D6	D5	D4	D3	D2	D1	D0	
		CHERCHER PISTE 0								
INSTRUCTION	W	0	0	0	0	0	1	1	1	Codes d'instruction
	W	X	X	X	X	X	HD	US1	US0	
EXECUTION										Replacer tête sur piste 0

L'INSTRUCTION SEEK, RECHERCHE D'UNE PISTE

Les lecteurs de disquette déplacent la tête de lecture/écriture avec ce qu'on appelle des moteurs de pas. Ces moteurs ne sont pas sans cesse en mouvement mais sont déplacés par des impulsions selon des changements d'angle bien définis par rapport à l'axe. Ces impulsions sont en règle générale produites dans les lecteurs de disquette eux-mêmes par des branchements digitaux parfois très onéreux. Un lecteur de disquette dispose de deux connexions vers l'extérieur. Sur une connexion sont placées des impulsions qui commandent le moteur de pas de façon à ce que la tête soit déplacée exactement d'une piste chaque fois qu'une impulsion se produit. La deuxième entrée sur les lecteurs détermine la direction de ce déplacement.

Ce déplacement de la tête est commandé par l'instruction Seek. Le FDC dispose en tout de quatre registres internes dans lesquels il stocke la position de tête actuelle des quatre lecteurs possibles. Ces registres sont sur zéro, ils sont donc annulés après l'instruction Recalibrate. Si une instruction Seek est envoyée à un lecteur déterminé, le contenu du registre de position correspondant est comparé à la valeur demandée. Si les deux valeurs sont identiques, aucune autre action n'est nécessaire.

Si cependant une différence entre ces deux valeurs est constatée, la polarité du signal DIR sur le pin 38 sera modifiée en fonction de la direction nécessaire et des impulsions de pas seront produites sur le pin 37. Le délai entre les impulsions de pas est programmable dans des limites assez larges avec l'instruction 'Indiquer lecteurs'.

Pour cette instruction et pour l'instruction Recalibrate, il n'y a pas de phase résultat. Il est recommandé que le programmeur entre toujours l'instruction 'lire état d'interruption' après une instruction Seek afin de terminer correctement l'instruction. Cette instruction fournit l'état du ST0, aussi appelé état d'interruption, dans la phase résultat. Sans cette instruction, le FDC n'accepte plus d'instructions de lecture/écriture.

PHASE	R/W	BUS DE DONNEES D7	D6	D5	D4	D3	D2	D1	D0	REMARQUES
						CHERCHER PISTE				
INSTRUCTION	W	0	0	0	0	0	1	1	1	Codes d'instruction
	W	X	X	X	X	X	HD	US1	US0	
	W	-----Numéro de piste-----------								
EXECUTION										La tête est placée sur le piste recherchée de la disquette

SENSE INTERRUPT STATUS, INTERROGER REGISTRE D'ETAT 0
Des interruptions sont produites par le FDC en exploitation NON DMA lors des évènements suivants:

pendant la phase exécution,
au début de la phase résultat,
à la fin d'un Seek ou d'un Recalibrate,
lors de la modification du signal Ready d'un lecteur

Si les deux premières causes d'interruption peuvent être aisément reconnues par le processeur, pour les deux autres causes d'interruption suivantes l'instruction 'Sense Interrupt Status' doit par contre être exécutée pour déterminer la cause de l'interruption. Les bits de la valeur du ST0 permettent de déterminer aisément la source d'interruption.

PHASE	R/W	BUS DE DONNEES D7	D6	D5	D4	D3	D2	D1	D0	REMARQUES
		INTERROGER ETAT D'INTERRUPTION								
INSTRUCTION	W	0	0	0	0	0	0	0	0	Codes d'instruction
RESULTAT	R	-----------Etat 0-------------								Information d'état à la fin de l'opération de recherche à travers FDC
	R	Numéro de piste après instruction de recherche								

SENSE DRIVE STATUS, INTERROGER ETAT LECTEUR

Cette instruction constitue la seule possibilité de déterminer le contenu du registre d'état ST3. Ce registre indique l'état du lecteur sélectionné. Il peut être lu à tout moment avec cette instruction.

PHASE	R/W	D7	D6	D5	D4	D3	D2	D1	D0	REMARQUES
		BUS DE DONNEES								
		INTERROGER ETAT LECTEUR								
INSTRUCTION	W	0	0	0	0	0	1	0	0	Codes d'instruction
	W	X	X	X	X	X	HD	US1	US0	
RESULTAT	R	-----------Etat 3------------								Information d'état à travers FDD

L'INSTRUCTION SPECIFY, INDIQUER DONNEES LECTEUR

Bien que cette instruction soit placée à la fin de notre brève présentation des instructions du FDC, c'est la première instruction qu'il faut utiliser après un Reset ou après la mise sous tension du FDC. A l'aide de cette instruction, les lecteurs les plus divers peuvent être adaptés au FDC. Tous les délais nécessaires sont indiqués au FDC avec cette instruction sur 3 octets. Outre les délais d'attente, un bit décide si le FDC doit travailler en mode DMA ou en mode d'interruption.

Mais examinons les données du lecteur de disquette. Il y a d'abord ce qu'on appelle le Step Rate Time. Le FDC attend automatiquement l'écoulement de ce délai entre les différentes impulsions de pas. Comme les différents lecteurs fabriqués attendent des délais d'attente entre deux impulsions complètement différents, ce délai peut être adapté précisément aux valeurs nécessaires.
Le deuxième délai qui peut être fixé est le délai d'attente dont le FDC attend automatiquement l'écoulement après activation du signal Head Load. Ce délai d'attente appelé Head Load Time n'a de sens que pour les lecteurs 8", sur les plus petits lecteurs, la tête est presque toujours chargée avec le signal Motor On.
Le troisième délai pouvant être fixé est le Head Unload Time. Ce délai programmable est celui qui doit s'écouler, après un accès à la disquette, avant que le signal Head Load du FDC ne redevienne inactif. Ce délai également n'a de sens que sur les lecteurs de disquette qui effectuent réellement la commande de la tête à travers la connexion du FDC que nous avons indiquée.

PHASE	R/W	BUS DE DONNEES D7	D6	D5	D4	D3	D2	D1	D0	REMARQUES
		INDIQUER DONNEES LECTEUR								
INSTRUCTION	W	0	0	0	0	0	0	1	1	Code d'instruction
	W	--Step rate--> <élever tête---								Information d'état à la fin de l'opération de recherche à travers FDC
	W	---Charger tête --> pas DMA---								

Step rate pour 5 1/4"					
Bit	7	6	5	4	Délai
	0	0	0	0	32 ms
	0	0	0	1	30 ms
	0	0	1	0	28 ms
			.		.
			.		.
			.		.
	1	1	0	1	6 ms
	1	1	1	0	4 ms
	1	1	1	1	2 ms

Elever tête					
Bit	3	2	1	0	Délai
	0	0	0	0	0 ms
	0	0	0	1	32 ms
	0	0	1	0	64 ms
		.			.
		.			.
		.			.
	1	1	0	1	416 ms
	1	1	1	0	448 ms
	1	1	1	1	480 ms

Charger tête									
Bit	7	6	5	4	3	2	1	0	Délai
	0	0	0	0	0	0	0	0	4 ms
	0	0	0	0	0	0	0	1	8 ms
	0	0	0	0	0	0	1	0	12 ms
				.					.
				.					.
				.					.
				.					.
	1	1	1	1	1	1	0	1	500 ms
	1	1	1	1	1	1	1	0	504 ms
	1	1	1	1	1	1	0	1	508 ms

BIT DMA
1 = Mode DMA
0 = Mode Non DMA

3.1.2.3 LES REGISTRES D'ETAT DU FDC 765

Comme nous l'avons déjà indiqué, le 765 dispose de 5 registres d'état internes. Le registre d'état principal peut être lu à tout moment. Le contenu des registres d'état 0 à 2 est fourni à la fin des instructions de lecture et d'écriture. Il n'est pas possible d'accéder de façon sélective aux registres 1 et 2. Seul le contenu des registres 0 et 3 peut ^etre lu de façon sélective grâce aux instructions particulières correspondantes.

Les noms et abréviations spéciaux utilisés dans la description suivante sont tirés de la fiche technique NEC du 765. Malheureusement, même chez NEC, ces termes n'ont pas toujours été employés de façon systématique. C'est ainsi que le compte rendu d'application du 765 utilise parfois d'autres noms et abréviations que la fiche technique.

Avant que nous n'examinions maintenant le registre d'état en détail, nous allons encore expliquer brièvement une des notions utilisées. Le terme Cylinder ne signifie rien d'autre que track ou piste. Nous ne nous expliquons pas pour quelle raison les deux termes track et cylinder apparaissent parallèlement dans les fiches techniques. Nous avons essayé d'éviter autant que possible d'utiliser le terme cylinder. A certains endroits toutefois les abréviations utilisées n'auraient plus alors de sens. Dans ce cas, nous avons conservé la notion de cylinder.

LE REGISTRE D'ETAT PRINCIPAL

Dans ce registre sont représentées les principales données sur l'état actuel du FDC. C'est aussi avec ce registre qu'est réglé le hand shaking entre processeur et FDC. Les huit bits de ce registre indiquent les données et états suivants:

Bit 7 RQM : ReQuest for Master
Si ce bit est mis, le FDC est prêt à lire ou à envoyer un nouvel octet à travers le registre de données. Si par contre le RQM est à zéro, aucun transfert de données ne peut se faire pour le moment.

Bit 6 DIO : Data Input/Output
Si RQM indique qu'un transfert de données est possible, DIO signale la direction de données nécessaire. Un DIO mis signifie que le FDC a un octet de données pour le

processeur, un DIO annulé indique que le FDC attend un octet du processeur.

Bit 5 EXM : EXecution Mode
Ce bit n'est utilisé qu'en mode Non DMA. En exploitation DMA, ce bit est systématiquement annulé. En mode Non DMA, le bit EXM est mis lorsque la phase exécution a commencé. Au début de la phase résultat, EXM est à nouveau annulé. Ce bit permet donc de déterminer si les valeurs fournies sont des informations de secteur ou s'il s'agit des octets de la phase résultat.

Bit 4 CB : FDC Busy
Le bit CB mis signale que le FDC est en train de traiter une instruction de lecture ou d'écriture et qu'il ne peut recevoir d'autres instructions. A la réception du premier octet d'une chaîne d'instruction, ce bit est mis et il le reste jusqu'à ce qu'ait été lu le dernier octet de la phase résultat. Ce bit est ensuite automatiquement remis à zéro.

Bits 3-0 DB : FDD3-0 Busy
Ces quatre bits sont affectés aux quatre lecteurs possibles. Si une instruction Seek ou Recalibrate est lancée sur un de ces lecteurs, le bit correspondant du registre d'état principal sera mis. Dès qu'un de ces bits est mis, aucune instruction de lecture ou écriture ne peut être envoyée au FDC. D'autres instructions Seek ou Recalibrate restent cependant possibles pour les autres lecteurs de disquette. Le bit correspondant est alors mis pour chaque instruction supplémentaire.
Ces bits ne sont pas annulés automatiquement à la fin de l'instruction. Pour remettre à nouveau ces bits sur 0, l'instruction 'lire état d'interruption' doit être envoyée au FDC. Cette instruction efface les bits si l'instruction correspondante s'était déjà terminée.

LE REGISTRE D'ETAT 0

Le registre d'état 0 est également appelé registre d'état d'interruption car il indique en exploitation Non DMA la cause d'une interruption.

Bits 7,6 IC : Interrupt Code

Dans ces deux bits, le FDC fournit des indications sur le déroulement d'une instruction. Les deux bits donnent quatre possibilités:

Bit
7 6

0 0 Instruction terminée avec succès. C'est évidemment ce message que vous devez toujours souhaiter obtenir car il indique qu'un accès en lecture par exemple a réussi. Mais attention!

0 1 Instruction interrompue. Dans ce cas, l'instruction a bien été lancée mais elle ne s'est pas terminée avec succès. Ce message peut survenir par exemple pour des erreurs de lecture sur la disquette mais il est également envoyé sur le CPC après chaque lecture ou écriture d'un secteur. La raison en est le fait de renoncer au signal TC et la programmation ainsi rendue nécessaire du dernier secteur de piste de façon identique au secteur à lire. Lorsque survient cette combinaison, il n'y a donc pas absolument à considérer qu'une véritable erreur s'est produite.

1 0 Instruction non valable. L'instruction que vous avez indiquée n'a pu être lancée parce qu'il s'agit d'une instruction illégale.
Vous obtenez également ce message lorsque vous avez envoyé l'instruction 'Sense Interrupt Status' mais qu'il n'y a encore pour le moment aucune interruption.

1 1 Instruction interrompue. La cause de ce message est une modification du signal Ready du lecteur sélectionné, pendant une instruction. L'instruction a alors bien été commencée mais elle n'a pu être entièrement terminée. Vous obtenez par exemple ce message lorsque vous retirez la disquette du lecteur pendant un accès en lecture.

Bit 5 SE : Seek End

Dès qu'une instruction a été terminée, le FDC met ce bit sur 1.

Bit 4 EC : Equipment Check
Ce bit d'état indique d'une part si le lecteur de disquette annonce une erreur. En cas d'erreur, le bit EC est mis. La seconde cause pour un bit EC mis peut être l'absence du signal TRKO après un Recalibrate. Avec Recalibrate en effet, la tête est déplacée vers la piste 0, soit jusqu'à ce que le sensor de piste 0 envoie un message au FDC, soit jusqu'à ce que 77 impulsions de pas aient été envoyées au lecteur de disquette. Ce cas peut se produire sur les lecteurs à 80 pistes lorsque la tête de lecture/écriture se trouve sur les pistes intérieures 78 à 80. Dans ce cas on peut simplement répéter l'instruction 'Recalibrate'.

Bit 3 NR : Non Ready
Lorsque le lecteur de disquette sélectionné communique lors d'une instruction de lecture ou d'écriture qu'il n'est pas prêt, ce flag est mis. La tentative d'accès à une deuxième tête qui n'existe pas d'un lecteur simple face a également pour effet de mettre ce bit.

Bit 2 HD : HeaD adress
Ce bit informe sur la tête sélectionnée au moment de l'évènement d'interruption.

Bits 1,0 US : Unit Select
Ces deux bits indiquent quel lecteur est actif au moment de l'interruption.

LE REGISTRE D'ETAT 1
Ce registre informe dans la phase résultat sur le déroulement de la phase exécution.

Bit 7 EN : ENd of track
Ce flag est mis par le FDC lorsqu'il tente un accès à un secteur après la fin programmée de la piste.

Bit 6 INUTILISE, TOUJOURS ZERO

Bit 5 DE : Data Error
Lors de l'écriture de données, le FDC produit

automatiquement une valeur de contrôle (check sum), d'après le principe du 'Cyclic Redundance Check', valeur qui est sauvegardée sur la disquette avec les données. Ces valeurs de contrôle sont également formées lors de la lecture des données. Elles sont alors comparées aux valeurs sauvegardées. Si le FDC constate une différence entre les deux valeurs de contrôle dans les champs de données ou dans les champs ID, le flag DE est mis.

Bit 4 OR : Over Run
Le transfert de données entre processeur et FDC en lecture ou en écriture doit se dérouler en un temps maximum déterminé. C'est ainsi que les données sont fournies au processeur en lecture avec des intervales de seulement 26 ms. Si le processeur, pour n'importe quelles raisons, ne peut tenir cette vitesse, il peut arriver qu'un nouvel octet soit prêt à être lu avant que le dernier octet n'ait pu être lu par le processeur. Dans un tel cas on parle d' Over Run et le bit OR est mis.

Bit 3 INUTILISE, TOUJOURS ZERO

Bit 2 ND : No Data
Ce flag peut être mis pour plusieurs raisons. Ce flag est mis lors de l'exécution d'une instruction d'écriture, de lecture ou de scanning lorsque le controller ne trouve pas le secteur indiqué.
Lors de l'exécution de l'instruction 'lire ID secteur', le flag ND est mis si le controller n'arrive pas à lire sans erreur un champ ID. Dans ce cas également, la cause de l'erreur est une erreur dans la valeur de contrôle.
La troisième cause possible apparaît en liaison avec l'instruction 'lire piste' lorsque le secteur de départ indiqué n'a pu être trouvé sur la piste.

Bit 1 NW : Non Writable
Si on constate, lors de l'exécution des instructions 'écrire secteur', 'écrire secteur effacé' ou 'formater piste' que la disquette est protégée contre l'écriture, ce flag est mis.

Bit 0 MA : Missing Adress mark
Ce flag est toujours mis lorsque le FDC n'a pu trouver, lors de la lecture ou de l'écriture de données, l'ID secteur au cours d'une rotation complète de la disquette. L'absence de la Data Adress Mark ou de la Data Adress Mark effacée est également signalée par la mise de ce bit. En outre et simultanément, le flag MD du registre d'état 2 est mis.

LE REGISTRE D'ETAT 2

De même que ST1, ST2 fournit des indications sur le succès ou l'échec d'une instruction.

Bit 7 INUTILISE, TOUJOURS ZERO

Bit 6 CM : Control Mark
Si le FDC trouve lors de la lecture de données ou lors d'une instruction Scan un secteur avec une Data Adress Mark effacée, il met ce bit.

Bit 5 DD : Data error in Data field
Comme pour le flag DE (Bit 5 de ST1), ce bit est mis lors d'erreurs CRC. Ce bit n'est toutefois mis que pour des erreurs dans les champs de données.

Bit 4 WC : Wrong Cylinder (Track)
Lors du formatage d'une piste doivent être indiqués pour chaque secteur le numéro de secteur, le numéro de piste, le numéro de tête et la taille du secteur. Ces données sont sauvegardées dans l'ID secteur et sont lues lors des instructions de lecture. Si le FDC constate alors une différence entre le numéro de piste lu et le numéro de piste indiqué, il met le flag WC.

Bit 3 SH : Scan equal Hit
Si une instruction Scan est lancée qui compare les informations de secteur avec les données fournies par le processeur, ce bit sera mis si les données sont effectivement identiques.

Bit 2 SN : Scan Not satisfied
Si le FDC ne trouve lors d'une quelconque instruction Scan aucun secteur qui corresponde de la façon demandée aux données indiquées, le bit SN est mis.

Bit 1 BC : Bad Cylinder
Ce flag a une signification semblable à celle du flag WC. Il est mis lorsque le numéro de piste lu dans l'ID ne coïncide pas avec celui indiqué dans l'instruction et que le numéro de piste lu dans l'ID est &FF.

Bit 0 MD : Missing adress mark in Data field
Si le FDC ne peut trouver la Data Adress Mark ou la Data Adress Mark effacée lors de la lecture de données, ce bit est mis.

LE REGISTRE D'ETAT 3, L'ETAT DU LECTEUR DE DISQUETTE
Le contenu de ce registre ne peut être transmis au processeur qu'avec l'instruction 'déterminer état lecteur'. Les bits de ce registre reflètent l'état du lecteur sélectionné dans l'instruction.

Bit 7 FT : FaulT
Ce flag reflète l'état du signal Fault qui existe sur certains lecteurs de disquette. Lorsque le lecteur dispose d'une telle connexion et que ce bit est mis, c'est qu'une erreur s'est produite dans le lecteur de disquette.

Bit 6 WP : Write Protected
Ce flag indique si la disquette placée dans le lecteur est protégée contre l'écriture. Un flag WP mis signifie qu'il n'est pas possible d'écrire sur la disquette.

Bit 5 RY : ReadY
Ce bit est utilisé pour déterminer l'état du canal Ready du lecteur de disquette. Un flag RY mis signale l'état 'Drive Ready'.

Bit 4 TO : Track 0
Si la tête de lecture/écriture du lecteur sélectionné se trouve sur la piste 0 au moment de l'instruction, le flag TO est mis.

Bit 3 TS : Two Side
Les lecteurs à double tête placent cette connexion sur la masse. Sur les lecteurs à simple tête, ce signal est par contre normalement high. L'état du bit TS permet au programme de déterminer quel type de lecteur est connecté.

Bit 2 HD : HeaD adress
Ce bit reflète l'état du signal Head select du FDC (pin 27).

Bits 1,0 : Unit Select
L'état de ces deux bits est identique aux niveaux des deux canaux US du FDC (pins 28 et 29).

3.1.2.4 L'EMPLOI DU FDC 765 SUR LE CPC

Malheureusement les développeurs du CPC sont loin d'avoir utilisé toutes les possibilités du FDC. C'est ainsi que deux lecteurs seulement peuvent être connectés au lieu des quatre possibles. L'exploitation de lecteurs à double tête n'est pas non plus possible car le signal HEAD SELECT, s'il est bien connecté, n'est cependant pas utilisé. Le sort du signal HEAD LOAD est encore pire puisqu'il n'est connecté nulle part. Ce défaut est cependant le plus facile à admettre puisqu'une exploitation de disquette 8" est d'une part sans intérêt pour l'utilisateur 'moyen' du fait des énormes dimensions physiques de ces lecteurs et qu'elle est d'autre part rendue impossible par d'autre détails dans les connexions du controller.
Malgré ces réserves, le controller a été très intelligemment construit pour le but recherché, l'exploitation sans problème de deux lecteurs 3". Avec une économie maximum d'électronique, un controller a été réalisé qui présente d'excellentes caractéristiques techniques.

Malgré l'esprit d'économie des développeurs, on n'a heureusement pas limité la fiabilité de l'appareil. On a ainsi adapté comme 'auxiliaire' au FDC 765 un composant qui arrache aux experts en électronique, pour le moins, une moue d'approbation. Nous pensons au séparateur de données intégré à 20 pôles SMC 9229 de l'interface du CPC 464. Sur le CPC 664, c'est, certainement pour des raisons de coût, le 'petit frère' de ce séparateur de données, le SMC 9216 à 8 pôles qui a été employé. Bien que ne disposant pas de toutes les possibilités du 9229, il est tout aussi fiable comme séparateur de données. Tous les signaux pour l'interface disquette du FDC, à l'exception du signal pour la mise en marche des moteurs du lecteur de disquette, sont produits par le FDC et le séparateur de données.

Bien que l'exploitation DMA représente la méthode la plus simple et la plus élégante pour connecter le disk controller, c'est une autre voie qui a été choisie, certainement pour des raisons de coût. Le processeur synchronise le transfert de données au vu du registre d'état principal. Les interruptions produites par le controller ne sont pas utilisées. Effectivement, la connexion d'interruption du FDC n'est pas branchée.

Le FDC est situé sur les adresses de port &FB7E et &FB7F. A la première adresse se trouve le registre d'état principal, la deuxième adresse appartient au registre de données.
Une troisième adresse est occupée par le Controller Board. Sur le port &FA7E se trouve un flip-flop à travers lequel les moteurs du lecteur de

disquette sont commandés. Si on écrit un 1 sur ce port (OUT &FA7E,1 en Basic), les moteurs de tous les lecteurs connectés sont mis en marche, par contre si on écrit un 0, tous les moteurs sont à nouveau arrêtés.

L'intéressant est que la connexion Terminal Count soit reliée au pin Reset. Les deux connexions sont branchées ensemble et reçoivent ensemble une impulsion positive en cas de Reset. La production d'une impulsion TC séparée n'est pas prévue dans le controller. Cela pose bien sûr la question de savoir comment est terminé un accès en lecture à la disquette. (Remarque: les développements suivants ne valent pas seulement pour la lecture des données mais également pour les instructions d'écriture et les instructions Scan.)
Habituellement, une impulsion TC est envoyée après lecture du secteur voulu. En l'absence de cette impulsion, un I/O multi-secteur est exécuté, ce qui veut dire que le controller lit des données jusqu'à ce qu'il rencontre le dernier secteur de la piste. Comme cependant le dernier secteur de la piste doit être programmé dans les neuf octets de l'instruction de lecture, les développeurs ont eu recours à une astuce. Ils fixent simplement le numéro du dernier secteur de façon à ce qu'il soit identique au numéro du secteur à lire. Après que toutes les données du secteur aient été lues, le FDC met automatiquement fin à la procédure de lecture et commence la phase résultat.
Cette façon de procéder entraîne toutefois un point qui doit être pris en compte lors de la programmation des routines du controller. Si un secteur est lu avec ce procédé, le registre d'état 0 indique en retour une erreur au système d'exploitation. Il s'agit de l'erreur 'Instruction lancée mais non terminée correctement', le bit 6 du ST0 est mis. La cause exacte de l'erreur se trouve dans le ST1; ici, le bit 7 est mis. Cela signifie en clair: fin de la piste atteinte, tentative d'accès à un secteur après la fin de la piste. Cette erreur doit être ignorée par le système d'exploitation sans toutefois que cela entraîne que d'autres erreurs éventuelles soient ainsi ignorées.
Le branchement réalisé rend impossible l'exploitation de lecteurs de disquette 8". Mais les routines de l'AMSDOS ne sont pas non plus conçues pour l'exploitation de lecteurs de disquette 8". D'ailleurs il n'est pas ou très difficilement possible sous AMSDOS de connecter des lecteurs ayant des caractéristiques techniques différentes, comme par exemple 80 pistes ou double tête de lecture. La connexion de tels lecteurs est en principe possible. Le signal correspondant pour la sélection de la tête de lecture pour les lecteurs à double tête est même branché sur l'interface disquette. Il faut cependant alors réécrire des sections essentielles du DOS car l'AMSDOS ne supporte pas le signal HS.

3.2 LA STRUCTURE DE LA DISQUETTE

3.2.1 LES TROIS FORMATS DE DISQUETTE

Comme vous le savez, le CPC distingue trois différents formats de disquette, le format CPC standard, le format de données et le format IBM. Vous pouvez sélectionner le format à utiliser lors du formatage de la disquette en indiquant S ou V (pour Système et Vendor), D (pour Données) ou I (pour IBM). Nous allons maintenant examiner plus précisément comment les formats sont structurés et en quoi ils se distinguent les uns des autres.

Lors de chaque accès à une disquette, AMSDOS détermine automatiquement le format. Cela ne se produit cependant que lorsqu'aucun fichier n'est ouvert sur la disquette. Cette limite peut cependant parfaitement être acceptée puisque, comme vous le savez, les disquettes avec des fichiers ouverts ne doivent pas être retirées du lecteur de disquette. Nous avons déjà évoqué le critère de distinction qui est constitué par les numéros de secteurs différents correspondant à chaque format.
En fixant certaines cases mémoire de la Ram, on dispose d'une autre possibilité pour interdire ce log-in automatique, autre terme pour désigner la déterminatlon des paramètres dlsquette. Vous trouverez ces cases mémoire dans la liste de la Ram disquette, dans un chapitre ultérieur.

Un certain nombre de choses sont communes aux trois formats. Il y a d'abord le nombre des pistes sur la disquette. Ce nombre est toujours de 40, les différentes pistes étant numérotées de 0 à 39. La taille d'un secteur est également identique pour les trois formats: 512 octets par secteur. En outre, pour tous les formats, 64 fichiers au maximum peuvent ̀etre placés sur les disquettes.
Mais c'est à cela que se résument les points communs.

3.2.1.1 LE FORMAT CPC STANDARD OU FORMAT SYSTEME

Ce format est certainement le format de disquette habituel et le format le plus utilisé sur le CPC. Dans ce format, une piste contient 9 secteurs avec les numéros de secteur &41 à &49. D'autre part les deux premières pistes de la disquette sont réservées pour CP/M. Si donc vous voulez travailler sous CP/M, c'est ce format que vous devriez utiliser car ce

n'est qu'avec ce format qu'un lancement ou un Warm Boot (Control C) de CP/M est possible.

Les pistes réservées sont affectées comme suit:

Piste 0, Secteur &41: Secteur Boot.
Piste 0, Secteur &42: Secteur de configuraton
Piste 0, Secteurs &43 à &47: inutilisés.
Piste 0, Secteurs &48, &49 et
Piste 1, Secteurs &41 à &49: CCP et BDSOS.

Bien que nous ayons jusqu'ici parlé de trois formats, ce qui est fondamentalement exact, le programme 'FORMAT.COM' ne connaît cependant pas que les trois indications S, D ou I mais également une quatrième possibilité, le 'V'. Avec cette indication, la disquette est formatée comme avec S mais les pistes système ne sont pas écrites. Cette option est essentiellement conçue pour l'exploitation commerciale des logiciels CP/M qui doivent être vendus, pour des raisons de Copyright, sans CP/M. L'utilisateur doit écrire lui-même CP/M sur les pistes système après avoir acquis un de ces logiciels. On peut utiliser le programme SYSGEN.COM à cet effet.

3.2.1.2 LE FORMAT DE DONNEES

Dans ce format également, 9 secteurs sont formatés sur chacune des 40 pistes. Les numéros de secteur sont cependant &C1 à &C9. Avec ce format, il n'y a pas de pistes système réservées, de sorte que vous disposez avec ce format de 9216 octets supplémentaires sur la disquette.

3.2.1.3 LE FORMAT IBM

Ce format est identique au format de l'IBM-PC sous CP/M 86. Si par pur hasard vous disposez à côté de votre CPC d'une telle machine comme second ordinateur, vous pouvez lire et écrire les disquettes de l'IBM également sur votre CPC. Cela suppose toutefois que les deux ordinateurs travaillent avec la même taille physique de disquette. Le format IBM se traduit par une place mémoire un peu plus réduite que pour les deux formats précédents car 8 secteurs seulement sont formatés sur une piste. Ce nombre moindre de secteurs est cependant quelque peu compensé par le fait qu'une seule piste, la première piste, est réservée comme piste

système.

3.2.2 LA STRUCTURE DU CATALOGUE

Avec les trois formats, 64 fichiers maximum peuvent être sauvegardés sur la disquette. Dans cette section, nous allons examiner de plus près où et sous quelle forme les noms de fichier sont sauvegardés sur la disquette.

La structure fondamentale du catalogue est imposée par CP/M. Comme les disquettes doivent pouvoir être lues et écrites aussi bien sous CP/M que sous AMSDOS, l'AMSDOS a dû être adapté à la structure de catalogue de CP/M. Si donc nous parlons dans les développements suivants d'AMSDOS, toutes ces explications vaudront également pour CP/M, étant donnée la compatibilité entre les deux systèmes d'exploitation. Nous ne vous rappellerons donc pas chaque fois cette compatibilité.

La gestion automatique d'un catalogue constitue une des tâches essentielles de tout système d'exploitation de disquette. Seule l'existence d'un tel catalogue permet de trouver les données aussi rapidement sur la disquette. Si cependant ce catalogue ne contient que les noms de fichier, cela n'apporte pas encore l'accès rapide voulu. Dans ce cas en effet, il faudrait que vous vous occupiez vous-même de la bonne gestion des différents secteurs sur la disquette. Vous comprendrez la complexité d'une telle tâche s'il vous arrive un jour que le catalogue de votre disquette favorite devienne illisible et si vous devez alors sauver les données de la disquette 'manuellement'.
Le catalogue doit donc également comprendre, outre le nom du fichier, la situation physique des données sur la disquette. Ce sont précisément ces deux données qui sont sauvegardées par AMSDOS dans l'entrée catalogue d'un fichier. Avant cependant que nous n'examinions de plus près comment est faite cette sauvegarde, il est nécessaire que nous expliquions ici certaines notions que nous allons souvent employer par la suite.

Il y a d'abord la notion de SECTEUR. Un secteur est une zone qui est disposée pour les données sur la disquette, lors du formatage. Dans l'AMSDOS, les secteurs ont toujours une taille de 512 octets, d'autres systèmes d'exploitation utilisent par exemple des secteurs de 128, 256 ou même 1024 octets.
Lorsque des données doivent être lues sur la disquette, c'est toujours un secteur entier qui devra être lu. Il n'est pas possible de lire directement sur la disquette, de façon parfaitement sélective, uniquement

certains octets déterminés. Le secteur est donc la zone de données qui peut être appelée au niveau le plus bas.
Un ENREGISTREMENT est un bloc de données plus petit d'exactement 128 octets. Chaque secteur (d'AMSDOS) contient par conséquent exactement 4 enregistrements. Pourquoi cette subdivision? La raison tient à l'histoire de la genèse de CP/M. CP/M a été à l'origine développé avec des lecteurs de disquette 8". Sur ces lecteurs un secteur avait toujours, à l'époque du développement, une taille de 128 octets. Ce n'est que plus tard que des formats furent développés avec des secteurs de taille supérieure à 128 octets. Pour conserver la compatibilité avec le format utilisé précédemment, le secteur de plus grande taille fut subdivisé par le BIOS, par programme, en unités plus petites de 128 octets. La compatibilité était ainsi assurée. AMSDOS travaille également du point de vue logique au niveau de l'enregistrement. Cela signifie qu'AMSDOS et CP/M ne connaissent en fait absolument pas les secteurs.
Une troisième notion nécessite une explication. Il s'agit de la notion de bloc. Cette notion remonte également à la préhistoire de CP/M. Lorsqu'on veut stocker sur disquette des fichiers de plusieurs dizaines de K, il faut enregistrer un nombre considérable de secteurs occupés par le fichier. Ce nombre peut cependant être considérablement réduit si l'on réunit plusieurs enregistrements dans des blocs et qu'on enregistre les numéros de bloc. La taille des blocs peut être librement définie sous CP/M, les valeurs usuelles sont 1K (comme sous AMSDOS) ou 2K. Pour maîtriser les calculs nécessaires à cet effet, tous les secteurs libres, c'est-à-dire tous ceux qui ne sont pas occupés par les pistes système sont numérotés dans l'ordre, à partir des pistes inférieures.
Dans la pratique, sur une disquette en format S, le secteur &41 de la piste 2 contient les enregistrements 0 à 3 de la disquette. Sur le secteur &42 se trouvent les enregistrements 4 à 7, etc. Comme donc un bloc a sous AMSDOS une taille de 1K, il contient 8 enregistrements. Le bloc 0 se trouve donc sur les secteurs &41 et &42 de la piste 2, le bloc 1 occupe les secteurs &43 et &44 et le bloc 4 occupe par exemple le secteur &49 de la piste 2 ainsi que le secteur &41 de la piste 3.
Nous reconnaissons volontiers que ces calculs sont très compliqués mais ils sont tout simplement indispensables sous CP/M et AMSDOS. Mais revenons à notre catalogue.
Vous êtes-vous déjà demandé pourquoi même le plus petit programme prend toujours 1K sur la disquette, même s'il ne comporte qu'un octet et qu'il ne remplit donc ainsi même pas un enregistrement ni un secteur? Nous venons justement d'en découvrir la raison. Ce sont les numéros des blocs occupés par un fichier qui sont enregistrés dans le catalogue.

Examinons donc une entrée de catalogue de plus près. Chaque entrée occupe 32 octets. Commençons par les 16 premiers octets.

00 46 4F 52 4D 41 54 20 20 43 4F 4D 00 00 00 15 .FORMAT COM....

Il s'agit, comme pour l'entrée DIR, de l'entrée du programme FORMAT.COM. Voici ce qu'on peut dire sans connaissance approfondie d'AMSDOS. Le point entre le nom de fichier et l'extension n'est pas sauvegardé avec le nom puisque le nom de fichier est rempli par deux caractères espace (valeur ASCII 32 ou 20 hexa). Mais que signifie la valeur 0 devant le nom de fichier. Souvenez-vous du numéro user qui peut être affecté à chaque fichier. Ce numéro est rangé dans le premier octet de l'entrée et règle l'accès au fichier. Cette valeur a en outre une autre signification que nous étudierons un peu plus tard.
Le nom de fichier est suivi de trois autres octets zéro et d'un octet de valeur &15. La signification de ces octets n'est pas très simple à découvrir. Seuls les octets 12 et 15 importent mais nous ne voulons pas encore vous en révéler la signification. Examinons d'abord les 16 octets manquant encore de l'entrée.

55 56 577 00 00 00 00 00 00 00 00 00 00 00 00 00 UVW.............

La signification de ces octets n'est pas non plus très aisée à deviner. Toutefois, si nous essayions une instruction CAT alors que la disquette Master CP/M se trouve dans le lecteur, cela nous aiderait un peu à trouver la solution de ce problème. Dans le catalogue, le fichier FORMAT.COM est présenté avec une taille de 3 K. Comme 1 K correspond à un bloc et que trois des 16 octets ci-dessus contiennent une autre valeur que 0, on est en droit de supposer que ces numéros sont les numéros de bloc. Cette supposition est parfaitement fondée. Les blocs occupés par un fichier sont en effet enregistrés dans les octets 16 à 31 d'une entrée du catalogue.

Cela soulève beaucoup de questions nouvelles. La question principale est: que se passe-t-il lorsqu'un fichier dépasse 16 K? La réponse est très simple, AMSDOS offre simplement à de tels fichiers une entrée de fichier supplémentaire. Une telle extension d'entrée ne se distingue de la première entrée qu'à peu d'égards. C'est essentiellement que les autres blocs occupés sont enregistrés dans les octets 16 à 31. Le numéro user et le nom de fichier sont identiques dans les deux entrées.

La prochaine question qui se pose est: à quoi AMSDOS reconnaît-il qu'une

extension suit? Il nous faut pour cela connaître la signification de l'octet 15 d'une entrée. Un petit exemple de calcul est nécessaire. Notre exemple FORMAT.COM se compose de 3 blocs. Comme nous l'avons dit, un bloc contient 8 enregistrements. Le fichier FORMAT.COM peut donc comporter au maximum 24 enregistrements (18 hexa). Le 15 est en fait le nombre d'enregistrements du fichier FORMAT.COM en hexadécimal.
Si un fichier dépasse la taille maximum pouvant être couverte par une entrée, la valeur de l'octet 15 de l'entrée se calcule alors ainsi: 16 (blocs) * 8 (enregistrements)= 128 ou &80. Dès que cette valeur figure à cet endroit, AMSDOS considère automatiquement qu'une extension suit.
Pour qu'il n'y ait pas de confusion avec les très grands fichiers comportant plusieurs extensions, les extensions sont numérotées en ordre croissant dans l'octet 12, immédiatement après le nom de fichier. L'ordre dans lequel les extensions doivent être lues est ainsi déterminé. Avec cette organisation, un fichier peut atteindre une taille qui n'est limitée que par le fait qu'il ne reste plus de place dans le catalogue pour d'autres entrées ou par le fait que la disquette est pleine.

Mais revenons brièvement sur le premier octet de l'entrée. Comme nous l'avons déjà indiqué, le numéro user est enregistré dans cet octet.
Si vous avez déjà essayé le programme de lecture de secteurs quelconques ou le moniteur disquette et que vous avez ainsi inspecté les secteurs du catalogue de vos propres disquettes, vous avez peut-être rencontré pour certains fichiers la valeur &E5 au lieu d'un numéro user correct (0 à 15). Les noms de fichier correspondants n'apparaissent cependant pas dans le catalogue lorsque vous entrez l'instruction CAT. Ce n'est pas étonnant car il s'agit très probablement de fichiers que vous avez vous-même supprimés au moyen de l'instruction ERA.
AMSDOS est en effet très prudent dans la suppression des fichiers. Il recherche simplement le nom du fichier indiqué avec toutes ses extensions éventuelles et fixe le numéro user sur la valeur &E5. A partir de ce moment, ce fichier n'existe plus pour AMSDOS même si toutes les données figurent encore sur la disquette. Cette procédure prudente est très appréciable lorsque vous avez supprimé par erreur des fichiers importants. Il vous suffit en effet dans un tel cas de lire les secteurs du catalogue avec un moniteur disquette et de fixer à nouveau le numéro user sur une valeur raisonnable. Votre fichier sera alors ressuscité. La valeur &E5 comme marque de suppression n'a par ailleurs pas été choisie par hasard. Après le formatage, tous les secteurs contiennent la valeur &E5, y compris donc les secteurs du catalogue. Si un accès au catalogue est alors entrepris, AMSDOS trouve alors pour chaque entrée possible la marque de suppression sans qu'une procédure spéciale soit donc

nécessaire.

Cela nous amène au dernier point important du catalogue, la situation sur la disquette. Comme nous l'avons déjà indiqué, 64 fichiers peuvent être au maximum enregistrés dans le catalogue. A raison de 32 octets par entrée, on obtient un besoin de place mémoire de 2048 octets. Cela correspond à 2 blocs ou 4 secteurs ou encore 16 enregistrements. Pour les trois formats, les deux premiers blocs de la première piste libre sont systématiquement utilisés pour le catalogue. En format S, le catalogue se trouve donc sur les secteurs &41 à &45 de la piste 2, en format D sur les secteurs &C1 à &C5 de la piste 0 et en format I sur les secteurs &01 à &05 de la piste 1.
En guise de conclusion de ce chapitre sur le catalogue, encore deux conseils.
Si le bit 7 du premier caractère de l'extension du nom de fichier, c'est-à-dire du neuvième octet de l'entrée, est mis le fichier reçoit alors l'attribut READ-ONLY. Les fichiers ainsi marqués ne peuvent être ni supprimés ni renommés sous AMSDOS et CP/M. Il y a (au moins) deux moyens de mettre ce bit. La première possibilité est offerte par l'instruction CP/M transitoire STAT. La seconde possibilité est offerte par le moniteur disquette. Si vous avez trouvé l'entrée à protéger dans les secteurs du catalogue, additionnez la valeur 128 ou &80 à la valeur ASCII du premier caractère, modifiez cet octet en fonction du résultat et réécrivez tout simplement le secteur ainsi modifié sur la disquette.
Notre seconde astuce vous permettra de dissimuler un fichier aux regards non-autorisés. Vous devez pour cela, comme pour l'attribut READ-ONLY, fixer le bit 7 du second octet de l'extension. Cette tâche peut également ̂etre réalisée sous CP/M avec STAT ou sous AMSDOS avec notre petit moniteur disquette. Mais n'oubliez pas le nom d'un fichier que vous aurez ainsi protégé. Ni CAT ni DIR n'indiquerons plus en effet sa présence sur la disquette.

3.2.3 LA STRUCTURE DES FICHIERS

Après que le chapitre précédent nous ait donné un aperçu des informations enregistrées dans le catalogue, nous allons maintenant voir comment les données proprement dites figurent sur la disquette.

Se pose d'abord la question de savoir d'où les informations telles que type de fichier, longueur du fichier ou adresse de départ du programme chargé sont tirées lorsqu'on lit un fichier. Ces informations header ne figurent pas en effet dans l'entrée du catalogue. C'est donc qu'elles sont sauvegardées avec les données.

Cela pose cependant le problème de savoir sous quelle forme ces données supplémentaires doivent être sauvegardées pour qu'il n'y ait pas de confusion entre header et données. Nous allons effectuer quelques expériences qui nous donneront des indications sur la façon dont sont sauvegardées les données.

Entrez par exemple la ligne:

```
SAVE"X1.BIN",B,&1000,1024
```

sur votre CPC et appuyez sur la touche ENTER. Le contenu du fichier ne nous intéresse pas pour le moment et vous pouvez donc choisir n'importe quelle adresse comme valeur de départ. Ce qui importe par contre, c'est la longueur de la zone de mémoire sauvegardée. Comme nous l'avons vu auparavant, les fichiers sont toujours sauvegardés sous AMSDOS par blocs de 1K. Comme nous avons choisi une longueur d'exactement 1K, le catalogue devrait indiquer pour ce fichier une longueur de 1K. Pensez-vous! Le fichier occupe 2K.
Entrez à nouveau cette ligne mais en réduisant la valeur de la longueur de fichier de 100, à 924. A la surprise générale, le fichier ne semble pas être plus petit et il occupe toujours 2K.
Faisons une troisième tentative. Entrez comme longueur l'expression 1024-128. Et voilà, c'est accompli. Notre fichier X1.BIN n'occupe plus qu'1 K sur la disquette. Toute indication de longueur supérieure, ne serait-ce que d'un octet entraînera à nouveau une taille de fichier de 2 K.

Un enregistrement, autrement dit 128 octets, sont donc sauvegardés en plus des données. Dans cet enregistrement figurent toutes les informations header. L'enregistrement-header est le premier enregistrement de presque tous les fichiers. Pourquoi cette réserve?

Essayez par exemple le petit programme suivant:

```
10 OPENOUT"X1.DAT"
20 FOR i=1 TO 1024
30 PRINT#9,"x";
40 NEXT i
50 CLOSEOUT
```

D'après nos expériences précédentes, on pourrait supposer que le fichier X1.DAT figurera au catalogue avec 2K. Mais il n'en est pas ainsi. En fait le fichier n'a qu'une taille de 1K, soit exactement la taille correspondant au nombre de caractères qui ont été écrits dans le fichier.

Cela nous donne la règle suivante:

> Les purs fichiers ASCII qui ont été sauvegardés avec la marque de type de fichier &16, n'ont pas d'enregistrement-header. (Le système d'exploitation n'évalue que le quartet faible, c'est-à-dire que &16 et &36 sont également traités comme de purs fichiers ASCII)
>
> Sur tous les autres fichiers, donc par exemple les programmes Basic ou machine, le premier enregistrement est l'enregistrement-header, qui sera copié lors du chargement dans la zone header.

Bien. Mais comment un programme peut-il déterminer si un fichier est un pur fichier ASCII. En effet, il n'y a dans le catalogue aucune indication sur le type de fichier. Ce problème a certainement donné quelque peu la migraine aux programmeurs de l'AMSDOS.
Finalement on a choisi une voie qui semble à peu près passable.
Lors de l'ouverture d'un fichier d'entrée, une valeur de contrôle sur 16 bits est systématiquement constituée avec les 66 octets du premier enregistrement. Le résultat est comparé au contenu des octets 67 et 68 de l'enregistrement. Si les valeurs sont identiques, cet enregistrement est à peu près certainement un enregistrement-header. Si cependant cette condition de contrôle devait par hasard être remplie sur un fichier ASCII, le premier enregistrement du fichier serait perdu. Cette possibilité est cependant très peu probable, de sorte qu'elle n'est pas envisagée par le système d'exploitation.

3.2.4 L'ECRITURE PHYSIQUE DES DONNEES SUR LA DISQUETTE

Etes-vous de ceux qui aiment bien remonter au fond des choses? Dans ce cas ce chapitre est fait pour vous. Nous allons vous montrer comment les données figurent effectivement sur la disquette. Nous n'entendons pas par données, dans ce chapitre, le contenu de n'importe quels fichiers, mais tout ce qui est placé sur une piste, sous une forme ou sous une autre. Nous expliquerons également dans ce contexte des notions telles que ID secteur, Gap ou Adress Mark.

3.2.4.1 MF, MFM, BITS ET MAGNETISME

Comme vous le savez déjà, la surface de la disquette est subdivisée en 40 sections ou pistes. La tête de lecture/écriture du lecteur de disquette peut être placée de façon sélective au dessus de chacune de ces pistes au moyen d'un moteur de pas. De même qu'avec le lecteur de cassette, les données sont sauvegardées sur la disquette bit par bit. La sauvegarde bit par bit des données et l'utilisation d'enregistreurs de données magnétiques représentent cependant les seuls points communs entre les deux systèmes de sauvegarde.
Avec le lecteur de disquette, la couche magnétique de la disquette est entièrement magnétisée. Sur le lecteur de cassette, on aurait la plus belle surmodulation possible avec un facteur de résonance de 100 pour cent. Ce facteur de résonance 'idéal' n'est cependant absolument pas gênant pour le lecteur de disquette. Au contraire, plus la magnétisation est réussie lors de l'écriture, plus la lecture ultérieure des données sera simple.
Pour comprendre le format d'enregistrement utilisé sur le CPC, il nous faut faire un petit détour par la physique. La tête de lecture/écriture d'un lecteur de disquette se compose essentiellement d'une bobine. Les bobines sont parmi les composants électroniques les plus intéressants même si elles sont malheureusement souvent méconnues. Une des propriétés capitales des bobines est constituée par le fait qu'un champ magnétique dans une bobine produit une tension. Il est toutefois très important que le champ magnétique se modifie en permanence pour produire une tension continuelle dans la bobine. Un champ magnétique statique, qui ne se modifie donc pas, ne produit par contre aucune tension. On trouve ce principe par exemple dans la dynamo d'une bicyclette.
L'intéressant est que cette particularité peut être également inversée. Une tension placée sur une bobine produit un champ magnétique. Si une tension changeante est envoyée, un champ magnétique se modifiant au même

rythme sera produit.
Ces deux propriétés de la bobine sont utilisées dans la tête de lecture/écriture du lecteur de disquette. Si une information doit être écrite sur la disquette, une tension alternative est placée sur la tête qui produira un champ magnétique. Ce champ magnétique magnétise le revêtement de la disquette, la sauvegarde est ainsi effectuée. Dans le cas inverse, le champ magnétique alternatif sur la disquette produit dans la bobine une tension qui est amplifiée et traitée par une électronique appropriée.

On pourrait maintenant avoir l'idée de marquer un lowbit (bit 0) par une absence de magnétisation et un highbit (1) par contre par des champs magnétiques sans cesse changeants. On reconstituerait ainsi directement, lors de la lecture, les signaux digitaux. Ce n'est cependant pas tout à fait aussi simple. L'électronique et la mécanique du lecteur qui seraient en effet nécessaires avec cette méthode devraient en effet travailler de façon extrêmement précise car il faudrait mesurer de façon très exacte les durées pour low et high lorsque plusieurs low bits ou plusieurs high bits se suivent sur la disquette.
Le problème devient plus simple si on sauvegarde avec les données une fréquence fixe de référence. Cette fréquence permet alors de déterminer très aisément la longueur d'une cellule de bit. Mais l'enregistrement de champs magnétiques se modifiant rapidement dans le cas d'un high bit ne peut pas non plus être utilisée sous cette forme. Au lieu de cela, on procède simplement ainsi: si un high bit doit être sauvegardé, on produit une impulsion exactement entre deux impulsions de fréquence, alors que pour un zéro on ne produit aucune impulsion entre deux impulsions de fréquence. La figure 3.2.4.1.1 montre comment peut se présenter une telle chaîne d'impulsions.
Cette chaîne d'impulsions commande un flip-flop (bascule électronique) qui inverse l'état de sa sortie lors de chaque impulsion d'entrée. Vous voyez dans la figure 3.2.4.1.2 la tension de sortie qui en résulte.
Le signal de sortie résultant convient parfaitement à la commande de la tête de lecture/écriture. Si la tension de sortie du flip-flop est high, un courant électrique traverse la bobine dans une direction, la couche magnétique de la disquette est magnétisée dans une direction. Si cependant la tension de sortie du flip-flop se modifie, la direction du courant dans la bobine s'inverse également, ce qui entraîne également une inversion de la direction de magnétisation de la disquette.

1) IMPULSIONS DE FREQUENCE

2) HIGH-BIT

3) LOW-BIT, NON ENREGISTRE

4) UNE CELLULE DE BIT = 2 UNITES D'HORLOGE

E5 = 1 1 1 0 0 1 1 0

FIGURE 3.2.4.1.1

DEROULEMENT DU SIGNAL SUR LA TETE DE LECTURE/ECRITURE LORS DE L'ECRITURE (DENSITE SIMPLE)

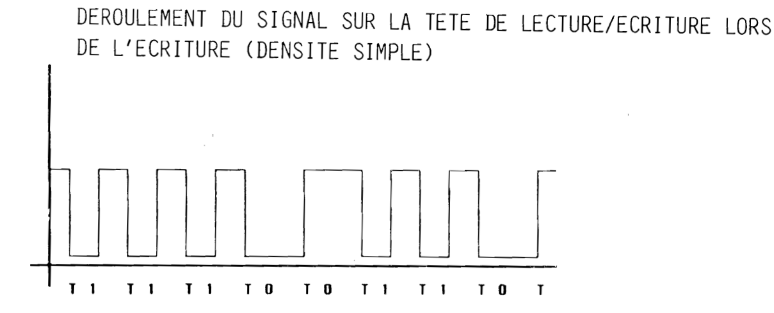


FIGURE 3.2.4.1.2

Si les données sont ainsi enregistrées, avec des différences low-high, la disquette contient ensuite plusieurs petits aimants qui sont magnétisés dans l'une ou l'autre direction, en fonction de l'enregistrement. La figure 3.2.4.1.3 représente graphiquement cette disposition des aimants les uns à la suite des autres.

FIGURE 3.2.4.1.3

Mais que se passe-t-il lors de la lecture des données? Nous avons dit que la disquette est entièrement magnétisée. Les nombreux petits aimants défilent devant la tête de lecture grâce à la rotation de la disquette. Chaque modification du champ magnétique produit dans la bobine une petite impulsion qui peut être amplifiée et traitée. Le reste du temps le champ magnétique ne se modifie pratiquement pas et aucune tension n'est donc produite. La figure 3.2.4.1.4 représente le signal de sortie amplifié. La ressemblance avec la première figure n'est pas le fait du hasard. Nous avons à nouveau lu entièrement les informations qui avaient été placées sur la disquette.

SIGNAL DE LECTURE RENFORCE

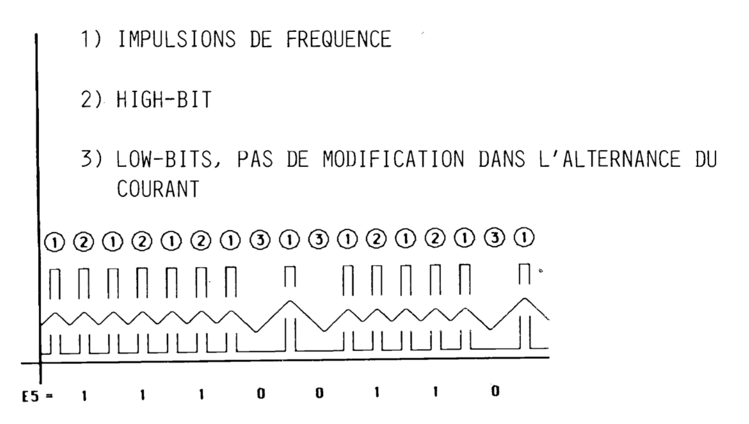


FIGURE 3.2.4.1.4

L'électronique d'évaluation n'a ainsi affaire, avec ce procédé, qu'à deux durées différentes. La première durée, la plus longue est celle entre deux impulsions de fréquence, elle marque les low bits. La seconde durée, la plus courte correspond au délai entre impulsion de fréquence et impulsion de données, elle marque les high bits. Comme les durées sont respectées avec une assez grande précision, les impulsions de données peuvent être séparées des impulsions de fréquence avec des branchements mono-flop simples puisqu'il y a toujours une fréquence disponible pour la synchronisation.

Encore un mot sur les durées qui apparaissent: une impulsion de fréquence dure toujours 500 nanosecondes (un demi millionième de seconde), le délai entre deux impulsions de fréquence est de 8 microsecondes. L'écriture ou la lecture d'un octet avec le procédé ainsi décrit nécessite donc 8 * 8 = 64 microsecondes. Avec une durée nominale de rotation de la disquette de 200 millisecondes, on peut donc placer sur la disquette 40 (pistes) * 200 (millisecondes) / 64 (microsecondes par octet) = 125000 octets.

Et pourtant vous disposez sur votre disquette d'environ 40000 à 50000 octets de plus? C'est vrai, nous vous avons présenté en premier le format en densité simple ou format MF. En partant du format que nous venons de décrire, nous allons maintenant décrire le format MFM ou format en double densité qui est utilisé sur le CPC.

Quelques 'cerveaux' n'étaient pas satisfaits du procédé que nous venons de décrire. On continua à fouiner jusqu'à ce qu'on développe un procédé, certes plus compliqué, permettant de presque doubler la capacité mémoire disponible. On partit pour cela de l'idée qu'il devait être possible de renoncer aux impulsions de fréquence, même si cela n'est pas entièrement possible. L'enregistrement des high bits ne crée pas de difficultés particulières puique chaque bit 1 à enregistrer produit sur la disquette une inversion de flux. Malheureusement, il faut également pouvoir enregistrer également des bits zéro. Et quand plusieurs bits zéro se suivent, la synchronisation devient très difficile.
La solution consiste à enregistrer simplement des bits de fréquence lorsque plusieurs zéros se suivent. Ce sont à nouveau les durées qui peremettent de distinguer entre les bits de données et les bits de fréquence. Il faut toutefois distinguer maintenant trois durées différentes. Les durées possibles sont représentées graphiquement dans les figures 3.2.4.1.5 et 3.2.4.1.6. La première figure représente les informations en entrée, la seconde représente l'enregistrement sur la disquette.

1) HIGH-BIT

2) LOW-BIT, pas d'enregistrement

3) BIT DE FREQUENCE, lorsque plus d'un low-bit doivent être enregistrés

4) 1 CELLULE DE BIT = 1 UNITE D'HORLOGE

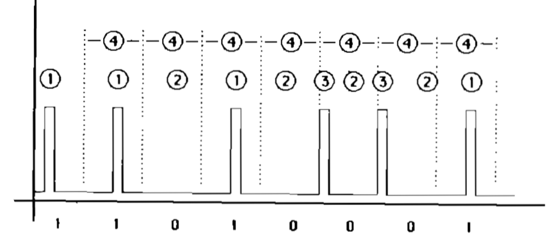


FIGURE 3.2.4.1.5

DEROULEMENT DU SIGNAL SUR LA TETE DE LECTURE/ECRITURE LORS DE L'ECRITURE (DOUBLE DENSITE)

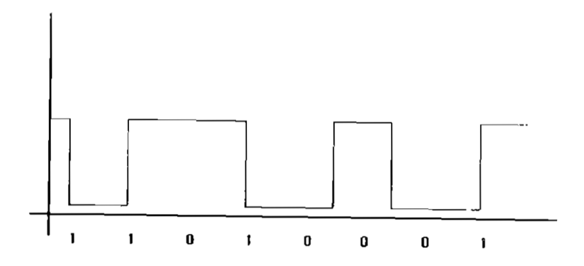


FIGURE 3.2.4.1.6

Une unité de temps simple est constituée par le temps qui s'écoule entre deux 1 consécutifs, une durée double caractérise la séquence de bits 1-0-1. Si plusieurs zéros se suivent, la troisième durée possible est utilisée. Cette durée d'une unité et demi signale le passage d'une impulsion de fréquence à une impulsion de données ou inversement.

Malheureusement les graphiques ne montrent pas très bien l'avantage immédiat du procédé MFM, du fait de l'échelle choisie. Essayez de vous représenter que la division du temps n'est pas la même dans les deux représentations graphiques. Si l'on se représente en effet que l'enregistrement physique avec ce procédé travaille avec le même nombre de changements de flux possibles, c'est-à-dire avec le même nombre de modifications possibles du champ magnétique, on obtient en fait un doublement de la capacité de la disquette.
Il est évident qu'avec ce procédé l'électronique d'évaluation doit être nettement plus précise qu'avec le format MF plus simple. Sur le CPC, l'évaluation est effectuée de façon extrêmement fiable par le séparateur de données dont nous avons déjà parlé.

Reste cependant encore une question. Comme une disquette formatée en format MF a une capacité de 125000 octets, une disquette écrite en format MFM ou format double densité devrait normalement pouvoir stocker 250000 octets. La capacité affichée n'est cependant que d'environ 182000 octets. Qu'est donc devenu le reste?

3.2.4.2 GAPs, IDs, et ADRESS-MARKS

Dans la section précédente, nous avons d'abord subdivisé la disquette en différentes pistes. Cela était rendu nécessaire par l'utilisation d'un moteur de pas pour la commande de la tête. Dans la présente section, nous allons diviser la disquette comme un gâteau, en différentes sections, les secteurs.
Si nous ne faisions pas cette subdivision, nous pourrions difficilement placer sur une piste plus d'un fichier. Et un fichier occuperait toujours une piste entière avec ses 6250 octets, quelle que soit la place effectivement nécessaire pour le fichier. Il s'agit là bien sûr d'un procédé très peu économe, surtout pour les fichiers de très petite taille. Pour la gestion des fichiers séquentiels, il faudrait en plus disposer dans l'ordinateur au moins un buffer d'une capacité de 6250 octets. Pour lire un fichier et écrire simultanément dans une autre fichier, le besoin de place mémoire s'élèverait même à 12500 octets. Il

s'agit d'une dépense de place mémoire considérable, même pour des ordinateurs d'une capacité de 64K.
Comme on voit, l'idée de subdiviser en zones plus petites est très intéressante. Mais pour séparer ces zones nettement entre elles, il faut effectuer un travail relativement complexe. Ce travail occupe de la place sur la disquette, de sorte que ce n'est plus la capacité originelle dont on disposera pour la sauvegarde des données. La première donnée est appelée la capacité non-formatée, la capacité effectivement disponible est appelée capacité formatée.
Nous allons maintenant examiner de plus près ce formatage et la capacité qui en résulte. La figure 3.2.4.2.1 représente les procédés décrits ici. Cette figure représente le contenu complet d'une piste.

La reconnaissance du début physique d'une piste ne pose pas de difficulté particulière au FDC, grâce à l'orifice d'index et à un faisceau lumineux bien réglé. Le faisceau lumineux d'index produit une impulsion lorsque le faisceau lumineux parvient au récepteur à travers l'orifice d'index. La fin de l'impulsion est pour le FDC le signal qui lui indique de commencer immédiatement le formatage. Son premier travail consiste à écrire sur la disquette 80 octets avec la valeur &4E. Cette zone est appelée GAP 4A.

Un GAP est, traduit littéralement, un vide. Pourquoi ce vide? La réponse est très simple. Le vide comble les tolérances entre les différents lecteurs de disquette. Malgré la haute précision incontestable des lecteurs de disquette, il y a malgré tout de petites différences dans le réglage des faisceaux lumineux d'index des différents lecteurs de disquette. Le GAP 4A a été choisi suffisamment grand pour que ces différences de réglage ne prêtent pas à conséquence, tant qu'elles restent dans certaines limites. C'est ainsi que les supports de données, c'est-à-dire les disquettes elles-mêmes peuvent être lues et écrites sans difficulté sur des lecteurs différents.
Après le GAP 4A est écrite une zone SYNC de 12 octets d'une valeur de 0. Comme nous l'avons vu, aussi bien en format MF qu'en format MFM, c'est la fréquence d'horloge qui est enregistrée lorsqu'il y a plusieurs 0 qui se suivent. Le terme SYNC signifie que le FDC est synchronisé avec la fréquence d'horloge. A la suite des 12 octets Sync, le FDC écrit trois octets qui sont appelés Index Adress Mark. Ces trois octets sont structurés d'après un schéma très spécial et ils peuvent être reconnus par le FDC grâce à une électronique spéciale du chip. L'Index Adress Mark n'est écrite qu'une fois au début d'une piste, c'est-à-dire immédiatement à la suite de l'orifice d'index. Elle constitue en quelque sorte une marque supplémentaire du début de la piste. A la suite de l'Index Adress

Mark est formaté un octet avec une valeur de &FC puis un autre GAP, le GAP 1, constitué de 50 octets &4E. Ce GAP est nécessaire pour donner au FDC, lors d'une lecture ultérieure, suffisamment de temps pour le traitement interne de l'Index Adress Mark. Le formatage du début de piste caractéristique est alors achevé. Ensuite viennent des zones qui existent pour chaque secteur.

Chaque secteur commence par 12 octets Sync. Comme précédemment, ce sont uniquement des zéros, donc uniquement les informations de fréquence, qui sont écrits. Les octets Sync sont suivis d'une Adress Mark, l'ID Adress Mark cette fois. L'ID AM a également 3 octets de long et elle peut être décodée électroniquement par le FDC. Après l'ID AM suit un octet d'une valeur de &FE.
Jusqu'à cet endroit, tous les secteurs de la disquette sont identiques. Mais pour pouvoir accéder plus tard de façon sélective aux pistes et secteurs, il est nécessaire de les marquer. C'est exactement cette tâche qui est prise en charge par les quatre octets qui suivent maintenant et qui constituent le champ ID. Dans l'ordre d'écriture sont entrés ici le numéro de piste, la face de disquette (0 ou 1), le numéro de secteur et la taille du secteur. La dernière indication ne peut bien sûr pas être entrée directement car il faudrait sinon un champ de deux octets pour les secteurs de 512 octets de long. Le codage correspondant correspond aux indications qui sont faites lors du formatage. Les secteurs de 512 octets de long sont marqués par un 2 dans cet octet.
On donne ainsi à chaque secteur son identification spéciale, l'ID secteur. Celle-ci permettra plus tard un accès sélectif. Les développeurs du système n'ont cependant rien laissé au hasard. Pour être protégé contre des erreurs éventuelles, une valeur de contrôle (checksum) de deux octets est formée d'après un procédé spécial à partir des trois octets de l'ID AM, de l'octet &FE et des quatre octets du champ ID. Le procédé utilisé est appelé Cyclic Redundancy Check ou CRC. Il permet une détection extrêmement fiable des erreurs de lecture. Les deux octets CRC ainsi obtenus sont placés immédiatement à la suite du champ ID.
Pour donner au FDC suffisamment de temps pour tester les octets ID et CRC lors d'une lecture ultérieure des données, vient ensuite d'abord un autre GAP, le GAP 2 d'une longueur de 22 octets. Sur ce GAP également, la valeur de données employée (qui n'a d'ailleurs pas d'importance) est à nouveau &4E. Le GAP 2 a une signification supplémentaire. Toutes les données d'un secteur seront écrites à la suite du GAP 2 lors de l'écriture ultérieure d'un secteur. Le GAP 2 permet au FDC d'effectuer la commutation entre lecture et écriture, une procédure qui prend toujours un certain temps.

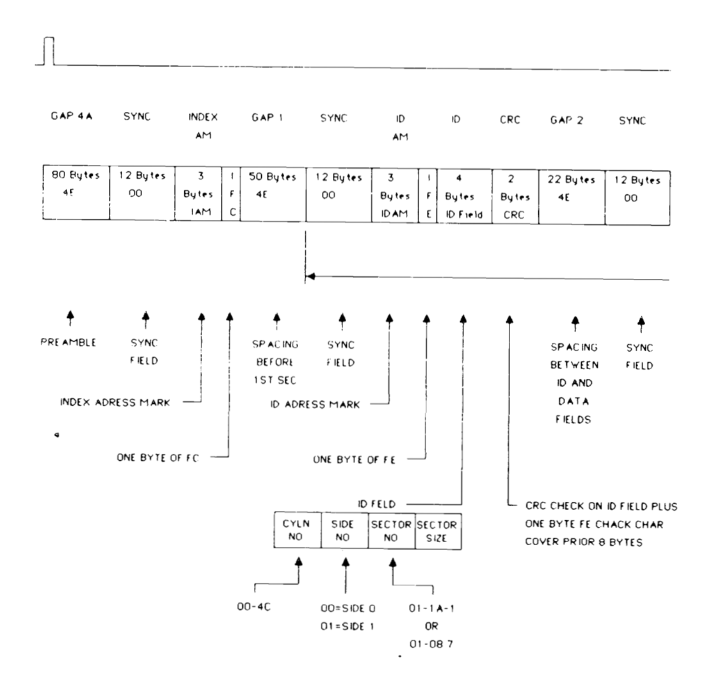
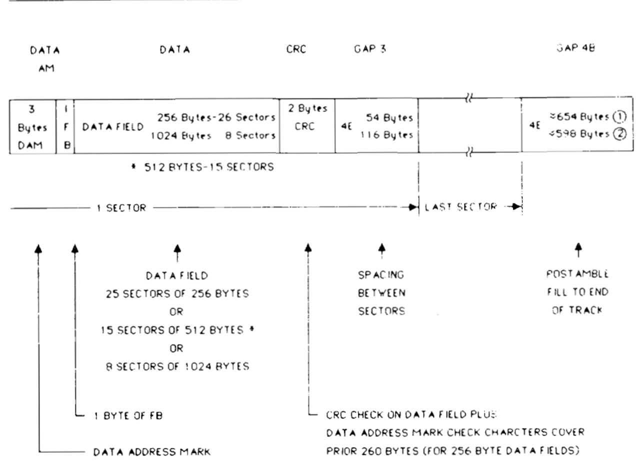


FIGURE 3.2.4.2.1





(1) FOR 256 BYTES/SECTOR
(2) FOR 1024 BYTES/SECTOR *
* 512 BYTES - 15 SECTORS
NOT AN IBM STANDARD
BUT OFTEN USED

Avez-vous deviné ce qui vient après le GAP 2? Exact, c'est une zone Sync de 12 octets zéro qui annonce enfin le début de la zone des données proprement dite. Mais les données ne sont pas encore écrites sur la disquette. Un Sync est en effet toujours suivi d'une zone Adress Mark. L'AM qui est maintenant formatée est la DATA Adress Mark qui a la même structure que l'ID AM et qui peut donc également être décodée par le FDC. Pour la distinguer de l'ID AM, la DATA AM est cependant suivie d'un octet d'une valeur &FB.
Et c'est maintenant que tout commence. Enfin vient la zone des données qui est remplie lors du formatage avec la valeur de l'octet de remplissage. La taille de cette zone est variable mais elle est fixée lors du formatage dans les indications qui sont fournies au FDC. C'est ainsi que sont possibles des zones de données d'entre 128 et 4096 octets. Dans ce dernier cas cependant, on ne peut placer qu'un seul secteur sur une piste, de sorte qu'on utilise presque toujours des tailles de secteur plus réduites. Sur le CPC, la taille de secteur est fixée à 512 octets.
Une valeur de contrôle de deux octets est également formée d'après le procédé CRC à travers la zone de données. Pour détecter également des erreurs dans l'ID DATA, ces octets sont inclus dans la formation du CRC, de sorte qu'avec une longueur de secteur de 512 octets, ce sont en tout 516 octets qui sont vérifiés. La fin d'un secteur est constituée par le GAP 3. La longueur de ce gap peut être indiquée lors du formatage. Sur le CPC, 82 octets &4E sont écrits lors du formatage. Ce gap a une signification tout à fait particulière. Si en effet de nouvelles données sont écrites plus tard sur les secteurs d'une disquette formatée, il est extrêmement improbable que la disquette tourne exactement à la même vitesse que lors du formatage. Or si la vitesse de rotation est ne serait-ce que légèrement supérieure, l'arc de cercle nécessaire pour le secteur sur la piste sera obligatoirement plus grand puisque la vitesse d'enregistrement est constante. S'il n'y avait pas le GAP 3, un tel enregistrement plus long pourrait tout simplement effacer un ID secteur venant immédiatement à la suite et le secteur suivant ne serait plus lisible. Les différences de longueur des secteurs produites par les variations de la vitesse peuvent cependant être neutralisées par le GAP 3 bien que le GAP 3 soit également produit lors de l'écriture d'un secteur. Cependant la longueur du GAP produit lors de l'écriture d'un secteur est considérablement réduite. En effet, 42 octets GAP seulement sont écrits, de sorte qu'il reste une marge suffisante pour les différences qui peuvent se produire.
Le secteur est alors entièrement formaté, le FDC demande maintenant les valeurs pour le prochain secteur et les écrit également sur la disquette. Quand tous les secteurs ont ainsi été formatés les uns à la suite des

autres, des octets GAP sont alors écrits jusqu'à ce que l'impulsion d'index apparaisse. Quand se produit l'impulsion d'index, le formatage de cette piste est terminé.

Comme on voit, le FDC a beaucoup à faire pour préparer une piste à être utilisable. D'autres FDC que le 765 sont loin d'être aussi prévenants. C'est ainsi par exemple que les disk controllers souvent utilisés de la série 197x ont par exemple besoin que presque chaque octet leur soit envoyé par le processeur, le programme de formatage devient alors très compliqué et long. Le soutien important qu'apporte le 765 transforme presque le formatage avec ce FDC en jeu d'enfant.

CHAPITRE 4: LA ROM ET LA RAM DE L'AMSDOS

Nous reconnaissons volontiers qu'il n'est pas donné à tout le monde d'étudier les détails internes d'un système d'exploitation. Beaucoup de possesseurs d'ordinateur sont totalement satisfaits si les programmes existants fonctionnent sans erreur et de façon fiable. Ces utilisateurs d'ordinateur peuvent en toute quiétude sauter le présent chapitre. Si cependant un jour les possibilités disponibles avec AMSDOS ne suffisent plus pour certaines tâches déterminées, le listing de la ROM peut être d'un très grand secours. La gestion de fichier relatif présentée dans cet ouvrage n'aurait par exemple pas pu être programmée sans une connaissance précise de la ROM. Pour tous ceux qui veulent abandonner les sentiers battus et se jeter dans le chaos des bits et des octets, une sorte de carte d'état-major peut se révéler extrêmement utile. Vous pouvez utiliser le listing de la ROM ainsi que les indications sur la RAM système en guise de carte d'état-major. Grâce à elle vous vous orienterez certainement plus aisément dans le système d'exploitation que si vous deviez chercher de nouvelles voies sans point de repère.

Nous regrettons les tâches blanches qui figurent encore sur cette carte d'état-major. Mais si vous étudiez le listing de façon approfondie, vous comprendrez peut-être que, même après deux mois de travail intensif sur le listing de la ROM, nous n'ayons pas pu élucider toutes les énigmes de l'AMSDOS. Le style de programmation des programmeurs de l'AMSDOS est parfois 'vraiment décapant'!

4.1 LA RAM SYSTEME DE L'AMSDOS

Comme vous le savez, après la mise sous tension du lecteur de disquette et du CPC, vous disposez d'une mémoire de 42249 octets pour les programmes Basic. Sans le lecteur de disquette, c'est-à-dire quand AMSDOS n'a pas été activé, ce sont cependant 43533 octets dont vous disposez comme le CPC vous l'indique si vous le lui demandez. La différence de 1284 octets n'est cependant pas entièrement à mettre à la charge du lecteur de disquette. Celui-ci se réserve cependant le plus grand morceau lorsqu'il est mis sous tension, soit 1024 octets. Cette zone de 1024 octets commence habituellement en &A700. Cette limitation a sa raison particulière dans la gestion des ROMs externes par le CPC. Le CPC peut gérer en tout 252 ROMs externes dans la zone de mémoire de &C000 à &FFFF. Il faut à cet égard distinguer trois différents types de ROMs, les ROMs de premier plan, les ROMs de second plan et les ROMs d'extension. C'est

le premier octet (&C000) de chaque ROM qui détermine le type d'une ROM donnée.
Après la mise sous tension, on teste d'abord si une ROM externe de premier plan se trouve sur le CPC avec l'adresse 0. On effectue dans ce but un OUT &DF00,0. Si le système d'exploitation constate qu'il 'se passe' quelque chose à cet endroit, la Rom externe est initialisée et prend ainsi le contrôle de la machine. Si cependant aucune Rom externe ne se trouve sur le port d'extension avec cette adresse de sélection de ROM, alors la ON Board ROM, le Basic, est initialisée comme ROM 0. Après l'initialisation, le programme de premier plan activé doit activer d'éventuelles ROMs de second plan. Chaque adresse de sélection de ROM, en commençant par l'adresse ROM 7 est alors interrogée à cet effet, pour savoir si une telle ROM de second plan existe. Dans le cas où existe une telle ROM, celle-ci est alors automatiquement initialisée.
Si lors de la mise sous tension de l'ordinateur une autre ROM que la ROM de premier plan intégrée a pris le contrôle du CPC, elle est alors libre de se réserver une zone de la RAM comme mémoire de travail. Elle peut par exemple décaler à cet effet vers le bas la limite supérieure de la RAM, c'est-à-dire le HIMEM. Lors de la réservation ultérieure de la mémoire pour AMSDOS, la mémoire occupée par AMSDOS sera alors également décalée. Dans ce cas, la RAM AMSDOS pourrait par exemple commencer en &A000.
Cette flexibilité de l'ordinateur a bien sûr également un inconvénient. Toutes les indications d'adresses que nous allons vous donner par la suite sur la zone commençant en &A700 n'ont pas de valeur absolue. Dans les conditions que nous venons de décrire, ces adresses peuvent en effet fort bien être situées dans une autre zone. Si l'on veut donc organiser des programmes avec accès direct à la RAM AMSDOS de la façon la plus souple possible, il ne faut accéder aux variables que de façon indirecte. On doit à cet effet rechercher le début effectif de la RAM AMSDOS et accéder aux variables avec le décalage correspondant au début de cette mémoire.
Ce qui semble un peu compliqué dans nos explications deviendra plus évident dans un exemple que nous vous donnerons un peu plus tard. Il nous faut cependant indiquer encore une autre particularité du lecteur de disquette ou de la RAM de l'AMSDOS. Il n'y a pas seulement, en effet, la zone de 1K que nous avons déjà décrite dans laquelle sont placées la plupart des variables système. Dans la zone de &BE40 à &BE7F, 64 autres octets sont utilisés pour le lecteur de disquette. Cette zone ne peut pas ^etre décalée, elle est fixée une fois pour toutes.
Cette zone est cependant quelque peu menacée. La pile du processeur est normalement placée de façon descendante à partir de &C000. Dans les programmes qui ne gèrent pas correctement cette pile, ou qui ont besoin

d'une très grande pile, par exemple à cause d'une programmation récursive, il peut parfaitement arriver que la pile devienne si grande, du fait d'un trop grand nombre de PUSHs ou de CALLs, qu'elle efface une partie de la zone mémoire du lecteur de disquette. Dans le premier cas, il faut corriger au plus vite le programme. Dans le second cas, il faut placer la pile dans une autre zone de la mémoire.

Mais venons-en maintenant à la liste de ces 64 octets, pour autant que nous en ayons compris la signification.

LISTE DE REFERENCE RAM SYSTEME BE40 à BE7F

BE40, BE41	Vecteur sur header paramètres disquette, lecteur A
BE42, BE43	Vecteur sur bloc paramètres disquette, lecteur A
BE44, BE45	Délai d'attente après MOTOR ON
BE46, BE47	Temps de fin de rotation du moteur du lecteur de disquette après le dernier accès
BE48	Valeur pour boucle d'attente lors du formatage d'une piste
BE49, BE4A	Valeurs pour une longue boucle de temporisation
BE4B	Nombre d'octets pour lecture de l'état d'interruption
BE4C-BE52	Buffer pour octets de la phase résultat du FDC
BE53	Lecteur (HS/US)
BE54	Piste
BE55	Enregistrement
BE56	Lecteur (HS/US)
BE57	Piste
BE58	Enregistrement
BE59	Nombre d'enregistrements par piste (masque de bloc + 1)
BE5A	Lecteur (HS/US)
BE5B	Piste
BE5C	Enregistrement
BE5D	
BE5E	Flag lire/écrire secteur
BE5F	Flag moteur marche/arrêt
BE60, BE61	Vecteur sur buffer I/O d'enregistrement
BE62, BE63	Vecteur sur buffer I/O de secteur
BE64, BE65	Mémoire provisoire de la pile

BE66	Nombre de tentatives de lecture
BE67, BE6C	Tick block
BE67, BE68	Ticker Chain, chaînage de la liste ticker
BE69, BE6A	Ticker count, nombre des tickers, pour expulser l'évènement
BE6B, BE6C	Reload Count
BE6D, BE73	Event block
BE6D, BE6E	Event Chain, chaînage des évènements
BE6F	Event Count
BE70	Event Class
BE71, BE72	Adresse de la routine EVENT
BE73	ROM select de la far adress pour la routine Event.
BE74	Buffer pour le numéro de piste voulu
BE75	Buffer pour code d'opération à envoyer au FDC
BE76, BE77	Vecteur sur buffer I/O de secteur
BE78	Flag pour messages d'erreur du controller ON/OFF
BE79, BE7C	Non-utilisé dans la version actuelle d'AMSDOS
BE7D, BE7E	Mémoire IY, low adress memory pool pour le lecteur de disquette
BE7F	Vecteur pour la manipulation des routines disque (DISC OUT OPEN etc.), normalement &C9, RETurn.

Il nous faut encore faire quelques remarques sur ces adresses. Comme il s'agit d'adresses absolues, comme nous l'avons déjà indiqué, et que leur situation dans la zone d'adresses ne change pas, il est possible d'intervenir ici par de simples POKEs. Toutefois certaines variables ne sont modifiées qu'au cours du déroulement de routines disque, de sorte qu'on ne peut y intervenir utilement à partir du Basic. Il est toutefois possible d'intervenir de façon importante en d'autres endroits par quelques POKEs.

Essayez par exemple de POKEr une valeur de 1 dans la case mémoire &BE45. Entrez ensuite l'instruction CAT. L'effet est surprenant. Si vous avez assez de patience, vous obtenez un CATalogue tout à fait normal. Cela aura toutefois duré beaucoup plus longtemps que d'habitude. Les valeurs des cases mémoire &BE44 et &BE45 déterminent en effet le temps d'attente

après la mise en marche du moteur. On attend normalement environ une seconde mais nous avons considérablement augmenté cette durée.

Vous pouvez obtenir par des POKEs dans les cases mémoire &BE46/&BE47 une autre manipulation de durées qui peut être très intéressante dans certains cas. Les valeurs de ces cases mémoire déterminent le temps de fin de rotation des moteurs du lecteur de disquette après le dernier accès à la disquette. Il peut cependant parfaitement arriver lors de la lecture et du traitement des données d'un fichier, que ce délai soit justement en train de s'écouler. Le moteur est alors à nouveau mis en marche pour le prochain accès mais on attend encore à nouveau une seconde pour que le moteur arrive à plein régime. Si le temps de fin de rotation est tellement allongé que le moteur ne s'arrête plus entre les différents accès, on peut constater un net accroissement de la vitesse de traitement. Essayez donc vous-même.

Les octets des adresses &BE4C à &BE52 sont également très intéressants. C'est ici que sont placées les indications de la phase résultat après toutes les opérations du FDC. Vous pouvez interpréter vous-même à votre guise ces valeurs, ce qui est extrêmement important dans le cas où on interdit les messages d'erreur du controller.

Nous arrivons maintenant à la prochaine case mémoire, &BE78, que nous avions déjà évoquée. Si une valeur de &FF est entrée ici, les messages d'erreur du FDC ne seront pas sortis sur l'écran. Dans ce cas il vous faut réagir vous-même aux erreurs, au vu des octets de la phase résultat.

Les cases mémoire &BE7D et &BE7E sont très importantes pour l'accès à la zone RAM décalable de 1K de l'AMSDOS. C'est ici qu'est rangée l'adresse de départ de cette zone de 1K. Ici figure normalement l'adresse &A700. Si cependant le début de la RAM AMSDOS est modifié pour une raison quelconque, c'est dans ces cases mémoire que vous trouverez l'adresse de départ correcte.

La case mémoire &BE7F est certainement très intéressante pour les grands experts. Vous trouvez ici normalement la valeur &C9. C'est le code d'opération Z80 pour RETurn. Chaque accès aux routines CAS détournées passe par ce RETurn. Vous avez ainsi une possibilité très simple pour intervenir aussi bien avant qu'après l'exécution des routines. Il est certes également possible de détourner encore une fois les routines CAS détournées, c'est-à-dire de leur fournir les adresses de vos propres routines, mais ce procédé est très compliqué.

Mais venons-en maintenant à la partie de loin la plus vaste de la Ram système. Dans la présentation suivante, nous sommes parti du principe qu'aucune extension n'est connectée. Nous nous référons ainsi aux adresses de la zone commençant en &A700.
Ceux d'entre vous qui sont des experts ès CP/M devront toutefois chercher un peu par eux-mêmes, l'organisation de la RAM nécessaire pour le BIOS est ici quelque peu différente. C'est ainsi que les FDCs figurent aux adresses standard (&5C à &7C). Les headers de paramètres de disque figurent dans la mémoire à partir de &AE58, les blocs de paramètres de disque à partir de &ADD8.

LISTE DE REFERENCE RAM SYSTEME A700 à AA80

A700	lecteur appelé
A701	user appelé
A702	lecteur actif
A703, A704	pointeur sur le header de paramètres de disque (A910/A920) lecteur actif
A705	flag si OPEN actif sur lecteur appelé
A706,A707	mémoire provisoire pour pointeur de pile pour toutes les routines logiques
A708,A72B	bloc de contrôle de fichier supplémentaire pour OPENIN
A709-A728	buffer bloc de contrôle de bloc (FCB) OPENIN
A708	flag pour OPENIN ff= pas d'OPENIN actif 00= OPENIN sur lecteur A 01= OPENIN sur lecteur B
A709	numéro user pour OPENIN
A70A-A714	nom de fichier pour OPENIN, 8 caractères nom de fichier, 3 caractères extension
A715	&00 première entrée sinon numéro de l'extension
A716	&00
A717	&00
A718	nombre d'enregistrements de cette extension
A719-A728	numéros des blocs de cette extension
A729-A72B	nombre d'enregistrements lus jusqu'ici pour ENTREE

A72C-A74F	bloc de contrôle de fichier supplémentaire pour OPENOUT
A72D-A74C	buffer bloc de contrôle de bloc (FCB) OPENOUT
A72C	flag pour OPENOUT ff= pas d'OPENOUT actif 00= OPENOUT sur lecteur A 01= OPENOUT sur lecteur B
A72D	numéro user pour OPENOUT
A72E-A738	nom de fichier pour OPENOUT, 8 caractères nom de fichier, 3 caractères extension, complété avec des espaces
A739	&00 première entrée sinon numéro de l'extension
A73A	&00
A73B	&00
A73C	nombre d'enregistrements de cette extension
A73D-A74C	numéros des blocs de cette extension
A74D-A74F	nombre d'enregistrements lus jusqu'ici pour SORTIE
A750-A799	header de fichier OPENIN
A750	1 = Disk In Char 2 = Disk In Direct
A751-A752	vecteur sur début du buffer OPENIN de 2K
A753-A754	vecteur sur caractère actuel dans le buffer OPENIN
A755	numéro user du fichier, élément du nom de fichier
A756-A764	nom de fichier pour le header de fichier, complété avec des zéros
A765	numéro de bloc
A766	dernier bloc
A767	type de fichier fichier ENTREE
A768-A769	longueur de données
A76A-A76B	emplacement de données
A76C	premier bloc
A76D-A76E	longueur logique
A76F-A770	adresse Entry
A771-A794	champs user libres pour l'utilisateur
A795-A797	compteur sur trois octets nombre de caractères lus
A798-A799	valeur de contrôle sur deux octets formée sur le header fichier OPENIN (A755-A797)
A79A-A7E3	header de fichier OPENOUT

A79A	1 = Disk Out Char 2 = Disk Out Direct
A79B-A79C	vecteur sur début du buffer OPENOUT de 2K
A79D-A79E	vecteur sur caractère actuel dans le buffer OPENOUT
A79F	numéro user du fichier, élément du nom de fichier
A7A0-A7AE	nom de fichier pour le header de fichier, complété avec des zéros
A7AF	numéro de bloc
A7B0	dernier bloc
A7B1	type de fichier fichier SORTIE
A7B2-A7B3	longueur de données
A7B4-A7B5	emplacement de données
A7B6	premier bloc
A7B7-A7B8	longueur logique
A7B9-A7BA	adresse Entry
A7BD-A7BE	longueur du bloc de données pour Disk Out Direct
A7BF-A7C0	adresse Entry pour Disk Out Direct
A7C1-A7DE	champs user libres pour l'utilisateur
A7DF-A7E1	compteur sur trois octets nombre de caractères écrits
A7E2-A7E3	valeur de contrôle sur deux octets formée sur le header fichier OPENOUT (A79F-A7E1)

A7E4-A8E3	buffer temporaire/buffer d'enregistrement Ce buffer est utilisé aussi bien comme buffer d'enregistrement que comme buffer pour vérifier et étendre le nom de fichier entré.
A874-A88A	buffer pour les vecteurs cassette qui sont à nouveau rétablis dans BC77... par l'instruction lTAPE
A88B-A88D	far address pour les vecteurs cassette détournés. Nécessaire pour l'emploi de RST 4. Pointe sur CD30 de la ROM 7.
A890-A8A8	bloc paramètres de disque supplémentaire lecteur A

A890, A891	SPT	enregistrements par piste (36)
A892	BSH	Block SHift (décalage de bloc) (3)
A893	BLM	BLock Mask (masque de bloc) (7)
A894	EXM	EXtend Mask (masque supplémentaire) (0)

A895, A896	DSM	nombre de blocs maximal (170)
A897, A898	DRM	nombre maximal d'entrées dans le catalogue -1 (63)
A899, A89A	ALO,1	taille catalogue (C000) codée en binaire, correspond à deux blocs
A89B, A89C	CKS	nombre des entrées à vérifier dans le catalogue (0010) 16 entrées
A89D, A89E	OFF	décalage de piste (2) pistes système occupées

A89F-A8A8 paramètres FDC

A89F	FSC	premier secteur de chaque piste (&41)
A8A0	PST	secteurs physiques par piste (9)
A8A1	GPS	longueur gap 3 pour lecture/écriture secteur (&2A)
A8A2	GPT	longueur gap 3 pour formatage piste (&52)
A8A3	FLB	octet de remplissage pour formatage piste (&E5)
A8A4	BPS	octets par secteur (2) correspond à 512 octets
A8A5	RPS	enregistrements par secteur (4)

A8A6	buffer pour piste actuelle
A8A7	flag pour chercher piste 0, read/write recalibrate
A8A8	flag, si login doit se produire à chaque accès disque

A8A9-A8B8	CSA	16 octets pour les valeurs de contrôle
A8B9-A8CE	ALT	22 octets table d'affectation, affectation blocs lecteur A

A8D0-A8E8 bloc paramètres de disque supplémentaire lecteur B affectation comme DPB lecteur A

A8E9-A8F8	CSA	16 octets pour les valeurs de contrôle
A8F9-A90E	ALT	22 octets table d'affectation, affectation blocs lecteur B

A910-A91F header paramètres disque lecteur A

A910-A911	XLT	table de conversion Skew Factor (inutilisée)
A912-A913	TRACK	mémoire BIOS piste actuelle. Attention! Est utilisé par AMSDOS comme DIRNUM

A914-A915	SECTOR	mémoire BIOS pour secteur actuel
A916-A917	DIRNUM	mémoire BIOS pour numéro DIR actuel
A918, A919	DIRBUF	pointeur sur buffer I/O de 128 octets (A930)
A91A, A91B	DPB	pointeur sur DPB lecteur A (A890)
A91C, A91D	CSV	pointeur sur mémoire pour formation valeur de contrôle (A8A9)
A91E, A91F	ALV	pointeur sur table d'affectation (A8B9)

A920-A92F header paramètres disque lecteur B

A910-A921	XLT	table de conversion Skew Factor (inutilisée)
A922-A923	TRACK	mémoire BIOS piste actuelle. Attention! Est utilisé par AMSDOS comme DIRNUM
A924-A925	SECTOR	mémoire BIOS pour secteur actuel
A926-A927	DIRNUM	mémoire BIOS pour numéro DIR actuel
A928, A929	DIRBUF	pointeur sur buffer I/O de 128 octets (A930)
A92A, A92B	DPB	pointeur sur DPB lecteur B (A8D0)
A92C, A92D	CSV	pointeur sur mémoire pour formation valeur de contrôle (A8E9)
A92E, A92F	ALV	pointeur sur table d'affectation (A8F9)

A930-A9AF	DIRREC	buffer de 128 octets pour un enregistrement catalogue. Est transféré du secteur DIR à ici

A9B0-ABAF	SECBUF	buffer pour le transfert physique de données vers et à partir du lecteur de disquette

Pour ces adresses également, cela vous 'démange' certainement d'observer l'effet de la manipulation de certaines de ces cases mémoire. Ne vous gênez surtout pas, il ne peut bien sûr rien arriver à l'ordinateur, comme vous le savez. Toutefois il n'est pas particulièrement recommandé de faire vos premiers essais en plaçant dans le lecteur, sans protection contre l'écriture, une de vos disquettes préférées. Une fois que les données qui figuraient sur une disquette sont détruites, il est trop tard pour penser à faire ce fameux backup que vous avez indéfiniment reporté à plus tard.

Si l'on examine précisément cette zone mémoire, on peut nettement dégager les limites de différentes sections. Nous avons doté ces sections de noms

qui ne sont cependant certainement pas connus de tous les lecteurs. Nous avons tiré ces noms de CP/M car beaucoup de sections de données ont des fonctions identiques à celles de sections comparables de CP/M. C'est ainsi que les sections Disc Parameter Header et Disc Parameter Block se présentent (presque) exactement sous cette forme dans CP/M. Tout ordinateur CP/M a des DPBs et des DPHs.
Les FDC, les File Control Blocks sont également utilisés dans CP/M. Toutefois certaines sections ont été étendues sous AMSDOS. Un DPB CP/M standard ne comprend que les 15 octets SPT à OFF, les extensions sont particulières à l'AMSDOS et ne peuvent être transférées sur d'autres ordinateurs CP/M, ni même sur le CP/M du CPC. Les header de fichier pour OPENOUT et OPENIN sont également particuliers à l'AMSDOS.

4.2 LE LISTING DE LA ROM DE L'AMSDOS

Vous trouverez sur les pages suivantes le listing de la ROM de l'AMSDOS. La ROM de l'AMSDOS contient toutes les routines nécessaires à l'exploitation du lecteur de disquette. Mais elle ne contient pas que cela. Comme nous l'avons déjà indiqué, l'AMSDOS n'occupe même pas la moitié des 16K disponibles dans la ROM. 8K entiers sont occupés par une partie du LOGO qui est fourni avec le lecteur de disquette. Cela divise donc par deux la zone disponible.
Cette zone LOGO n'a pas été imprimée dans le listing suivant, pour plusieurs raisons. D'une part c'est un livre sur le lecteur de disquette et non un livre sur LOGO que vous avez entre les mains. D'autre part la partie de LOGO figurant dans cette ROM n'est qu'une petite partie de l'interpréteur LOGO complet. Vous trouverez 32K supplémentaires de ce langage de programmation par ailleurs très intéressant sur la disquette système du CPC. Si donc nous imprimions et commentions la partie de LOGO située dans la ROM, personne ne pourrait en tirer quoi que ce soit car il s'agit d'une partie trop restreinte de l'interpréteur complet.

Mais même les 8K restants ne sont pas complètement utilisés par AMSDOS. 1024 octets de la zone de &DC00 à &DFFF ne sont absolument pas utilisés. Cette zone a été peut-être prévue pour des extensions ultérieures de l'AMSDOS. Nous n'avons plus maintenant que 7K qui ne sont cependant toujours pas entièrement à la disposition de l'AMSDOS. Des sections de CP/M sont également intégrées dans cette zone. CP/M et AMSDOS utilisent ainsi ensemble de nombreuses routines de la ROM alors que d'autres sont par contre exclusivement utilisées par CP/M ou par AMSDOS. C'est ainsi que sont intégrés dans la ROM deux programmes complets de commande pour deux interfaces sérielles qui peuvent être affectés sous CP/M de différentes façons aux différents périphériques à travers l'octet I/O.

Un examen du listing révèle qu'une fois retirées toutes les sections de mémoire qui sont exclusivement utilisées par CP/M, il ne reste qu'à peine 6K pour l'AMSDOS. Mais voyez plutôt par vous-même comment ces 6K sont construits.

```
*************** Préfixe pour ROM CPM
C000  01      DEFB   01H        ;ROM Type, Background ROM
C001  00      DEFB   00H        ;ROM Mark Number
C002  05      DEFB   05H        ;ROM Version Number
C003  00      DEFB   00H        ;ROM Modification Level

*************** Adresse de la table d'instructions
C004  72C0    DEFW   0C072H

*************** Bloc Jump instructions AMSDOS
C006  C3BCC1  JP     0C1BCH     ;CPM ROM
C009  C3B2C1  JP     0C1B2H     ;CPM
C00C  C3D1CC  JP     0CCD1H     ;DISC
C00F  C3D5CC  JP     0CCD5H     ;DISCIN
C012  C3E4CC  JP     0CCE4H     ;DISCOUT
C015  C3FDCC  JP     0CCFDH     ;TAPE
C018  C301CD  JP     0CD01H     ;TAPEIN
C01B  C318CD  JP     0CD18H     ;TAPEOUT
C01E  C3DACD  JP     0CDDAH     ;A:
C021  C3DDCD  JP     0CDDDH     ;B:
C024  C3E4CD  JP     0CDE4H     ;DRIVE
C027  C3FECD  JP     0CDFEH     ;USER
C02A  C32ED4  JP     0D42EH     ;DIR
C02D  C38AD4  JP     0D48AH     ;ERA
C030  C3C4D4  JP     0D4C4H     ;REN

*************** Bloc Jump instructions disc controller
C033  C372CA  JP     0CA72H     ;^81 enable/disable messages d'erreur
C036  C30DC6  JP     0C60DH     ;^82 indiquer données disque
C039  C381C5  JP     0C581H     ;^83 déterminer format disque
C03C  C366C6  JP     0C666H     ;^84 lire secteur
C03F  C34EC6  JP     0C64EH     ;^85 écrire secteur
C042  C352C6  JP     0C652H     ;^86 formater piste
C045  C363C7  JP     0C763H     ;^87 chercher piste
C048  C330C6  JP     0C630H     ;^88 déterminer état disque
C04B  C303C6  JP     0C603H     ;^89 fixer nombre tentatives lecture

*************** Bloc Jump entrées CP/M
C04E  C368C1  JP     0C168H
C051  C3DBC0  JP     0C0DBH
```

```
*************** Bloc jump routines I/O sérielles pour CP/M
C054  C389C3    JP      0C389H      ;initialisation complète SIO & 8253
C057  C301C3    JP      0C301H
C05A  C3DBC3    JP      0C3DBH      ;Canal A buffer RX plein?
C05D  C3F7C3    JP      0C3F7H      ;Canal A retirer un caractère
C060  C335C4    JP      0C435H      ;Canal A buffer TX vide?
C063  C345C4    JP      0C445H      ;Canal A envoyer un caractère
C066  C3E3C3    JP      0C3E3H      ;Canal B buffer RX plein?
C069  C3FFC3    JP      0C3FFH      ;Canal B retirer un caractère
C06C  C33AC4    JP      0C43AH      ;Canal B buffer TX vide?
C06F  C34BC4    JP      0C44BH      ;Canal B envoyer un caractère

*************** Table des instructions DOS
C072  43504D20  DEFM    'CPM RO','M'+80H
C076  524FCD
C079  4350CD    DEFM    'CP','M'+80H
C07C  444953C3  DEFM    'DIS','C'+80H
C080  44495343  DEFM    'DISC.I','N'+80H
C084  2E49CE
C087  44495343  DEFM    'DISC.OU','T'+80H
C08B  2E4F55D4
C08F  544150C5  DEFM    'TAP','E'+80H
C093  54415045  DEFM    'TAPE.I','N'+80H
C097  2E49CE
C09A  54415045  DEFM    'TAPE.OU','T'+80H
C09E  2E4F55D4
C0A2  C1        DEFM    'A'+80H
C0A3  C2        DEFM    'B'+80H
C0A4  44524956  DEFM    'DRIV','E'+80H
C0A8  C5
C0A9  555345D2  DEFM    'USE','R'+80H
C0AD  4449D2    DEFM    'DI','R'+80H
C0B0  4552C1    DEFM    'ER','A'+80H
C0B3  5245CE    DEFM    'RE','N'+80H

*************** Instructions Disc Controller
C0B6  81        DEFB    01H + 80H
C0B7  82        DEFB    02H + 80H
C0B8  83        DEFB    03H + 80H
C0B9  84        DEFB    04H + 80H
C0BA  85        DEFB    05H + 80H
C0BB  86        DEFB    06H + 80H
```

```
C0BC  87        DEFB   07H + 80H
C0BD  88        DEFB   08H + 80H
C0BE  89        DEFB   09H + 80H
C0BF  00        DEFB   00          ;Marque fin de la table

*************** sauver vecteur d'interruption et adresse de port GA
C0C0  2A3900    LD     HL,(0039H)  ;vecteur INT (RST 7)
C0C3  223EAD    LD     (0AD3EH),HL ;ranger dans la Ram
C0C6  3EC3      LD     A,0C3H      ;code d'opération JMP
C0C8  3233AD    LD     (0AD33H),A  ;pour module CALL
C0CB  AF        XOR    A
C0CC  3240AD    LD     (0AD40H),A
C0CF  F3        DI                 ;interdire INT pour utilisation
C0D0  D9        EXX                ;du jeu de registres alternatif
C0D1  ED433CAD  LD     (0AD3CH),BC ;sauver adresse port GA et config ROM
C0D5  D9        EXX                ;restaurer jeu de registres altern.
C0D6  21FAC0    LD     HL,0C0FAH
C0D9  181A      JR     0C0F5H

*************************************************************
C0DB  2140AD    LD     HL,0AD40H
C0DE  BE        CP     (HL)
C0DF  C8        RET    Z
C0E0  C5        PUSH   BC
C0E1  46        LD     B,(HL)
C0E2  77        LD     (HL),A
C0E3  B7        OR     A
C0E4  78        LD     A,B
C0E5  C1        POP    BC
C0E6  28E7      JR     Z,0C0CFH
C0E8  F3        DI                 ;nécessaire pour utiliser ancien jeu
C0E9  08        EX     AF,AF'      ;de registres
C0EA  D9        EXX
C0EB  ED4B3CAD  LD     BC,(0AD3CH) ;rechercher  adresse  port  GA  et  ROM-
                                   Select
C0EF  B7        OR     A           ;annuler Carry
C0F0  08        EX     AF,AF'
C0F1  D9        EXX
C0F2  2132C1    LD     HL,0C132H
C0F5  2234AD    LD     (0AD34H),HL ;Adresse du saut en AD33
C0F8  FB        EI                 ;autoriser à nouveau interruptions
C0F9  C9        RET
```

```
*************** module 'CALL AD33', conserver ancien jeu de registres
C0FA  F3        DI                   ;interdire interruptions
C0FB  08        EX     AF,AF'        ;pour travailler avec jeu de
C0FC  D9        EXX                  ;registres alternatif
C0FD  2238AD    LD     (0AD38H),HL   ;sauver HL
C100  E1        POP    HL            ;adresse de retour dans HL
C101  ED7336AD  LD     (0AD36H),SP   ;sauver pointeur de pile
C105  3100C0    LD     SP,0C000H     ;et initialiser
C108  D5        PUSH   DE            ;tous registres sur pile initialisée
C109  C5        PUSH   BC
C10A  F5        PUSH   AF
C10B  FDE5      PUSH   IY
C10D  ED4B3CAD  LD     BC,(0AD3CH)   ;(adresse de port Gate Array)
C111  B7        OR     A
C112  CD4FC1    CALL   0C14FH        ;chercher adresse après 'CALL AD33'
                                     ;et CALLer
C115  F3        DI                   ;pourrait être à nouveau autorisé
C116  08        EX     AF,AF'        ;échanger registres
C117  D9        EXX
C118  ED433CAD  LD     (0AD3CH),BC   ;a été éventuellement modifié
C11C  2163C1    LD     HL,0C163H     ;détourner à nouveau vecteur INT
C11F  223900    LD     (0039H),HL
C122  FDE1      POP    IY            ;restaurer registres
C124  F1        POP    AF
C125  C1        POP    BC
C126  D1        POP    DE
C127  2A38AD    LD     HL,(0AD38H)   ;hl n'avait pas été PUSHé
C12A  08        EX     AF,AF'
C12B  D9        EXX                  ;commuter sur jeu reg. standard
C12C  ED7B36AD  LD     SP,(0AD36H)   ;restaurer ancien pointeur de pile
C130  FB        EI
C131  C9        RET                  ;routine voulue exécutée

*************** module 'CALL AD33', ancien jeu reg. n'est pas sauvé
C132  F3        DI                   ;interdire INT
C133  08        EX     AF,AF'        ;jeu de registres alternatif
C134  D9        EXX
C135  E1        POP    HL            ;retirer adresse de retour de la pile
C136  ED7336AD  LD     (0AD36H),SP   ;sauver pointeur de pile
C13A  3100C0    LD     SP,0C000H     ;initialiser pile
C13D  CD4FC1    CALL   0C14FH        ;rechercher adresse après CALL AD33
                                     ;et JPer
```

```
C140  F3         DI                       ;pourrait être à nouveau autorisé
C141  D9         EXX                      ;commuter jeu de registres
C142  2163C1     LD      HL,0C163H        ;détourner à nouveau vecteur INT
C145  223900     LD      (0039H),HL
C148  D9         EXX                      ;commuter jeu de registres
C149  ED7B36AD   LD      SP,(0AD36H)      ;restaurer pointeur de pile
C14D  FB         EI
C14E  C9         RET                      ;routine voulue exécutée

*************** JP à l'adresse après 'CALL AD33'
C14F  ED5B3EAD   LD      DE,(0AD3EH)      ;(System Interrupt Vector)
C153  ED533900   LD      (0039H),DE       ;détourner vecteur INT
C157  FD2148AC   LD      IY,0AC48H        ;adresse de base pour Ram disque
C15B  5E         LD      E,(HL)           ;retirer octets après CALL AD33
C15C  23         INC     HL               ;de DE
C15D  56         LD      D,(HL)
C15E  D5         PUSH    DE               ;sur la pile
C15F  08         EX      AF,AF'
C160  D9         EXX
C161  FB         EI
C162  C9         RET                      ;et appeler à travers RET

***************
C163  CD33AD     CALL    0AD33H
C166  3800                                ;saut au vecteur INT

***************
C168  223AAD     LD      (0AD3AH),HL      ;ranger hl
C16B  E1         POP     HL               ;adresse RET dans hl
C16C  E5         PUSH    HL
C16D  23         INC     HL               ;augmenter de 2, donc après
C16E  23         INC     HL               ;indication de l'adresse
C16F  E3         EX      (SP),HL          ;échanger avec adresse RET
C170  E5         PUSH    HL               ;valeur originelle sur pile
C171  2A3AAD     LD      HL,(0AD3AH)      ;restaurer hl
C174  C333AD     JP      0AD33H

*************** entrer vecteur INT système, JP (DE)
C177  2163C1     LD      HL,0C163H
C17A  223900     LD      (0039H),HL
C17D  EB         EX      DE,HL
C17E  E9         JP      (HL)
```

```
*************** bloc de Jump BIOS, sous CP/M en Ram à partir de OAD00H
C17F  C3B2C1   JP     0C1B2H      COLD BOOT
C182  C3BEC2   JP     0C2BEH      WARM BOOT
C185  C3E1C2   JP     0C2E1H      CONSOLE STATUS
C188  C3C3C2   JP     0C2C3H      CONSOLE INPUT
C18B  C3C8C2   JP     0C2C8H      CONSOLE OUTPUT
C18E  C3D2C2   JP     0C2D2H      PRINTER OUTPUT
C191  C3D7C2   JP     0C2D7H      PUNCHER
C194  C3DCC2   JP     0C2DCH      READER
C197  C3E9C2   JP     0C2E9H      TRACK 0
C19A  C3F2C2   JP     0C2F2H      SELECT DRIVE
C19D  C324C5   JP     0C524H      SELECT TRACK
C1A0  C329C5   JP     0C529H      SELECT SECTOR
C1A3  C31AC5   JP     0C51AH      INSTALL BUFFER
C1A6  C3F7C2   JP     0C2F7H      READ SECTOR
C1A9  C3FCC2   JP     0C2FCH      WRITE SECTOR
C1AC  C3CDC2   JP     0C2CDH      PRINTER STATUS
C1AF  C35AC5   JP     0C55AH      TRADUIRE NUMERO SECTEUR

*************** CPM-COLD BOOT
C1B2  CD12B9   CALL   0B912H      ;KL CURR SELECTION
C1B5  4F       LD     C,A         ;ROM Selection
C1B6  21DCC1   LD     HL,0C1DCH   ;Entry Point Adresse
C1B9  C316BD   JP     0BD16H      ;MC START PROGRAM

*************** CPM ROM
C1BC  3806     JR     C,0C1C4H
C1BE  CD12B9   CALL   0B912H      ;KL CURR SELECTION
C1C1  B7       OR     A           ;tester si adresse ROM Select = 0
C1C2  2818     JR     Z,0C1DCH    ;=> LK 1 sur Contr.Board ouvert, CP/M
C1C4  FDE5     PUSH   IY          ;Adresse himem
C1C6  D5       PUSH   DE          ;Adresse lomem
C1C7  1100FB   LD     DE,0FB00H   ;hl = himem
C1CA  19       ADD    HL,DE       ;diminué de 0400h
C1CB  E5       PUSH   HL          ;ranger nouvelle himem
C1CC  23       INC    HL
C1CD  E5       PUSH   HL
C1CE  FDE1     POP    IY          ;iy = himem+1
C1D0  CDDDC5   CALL   0C5DDH      ;initialiser FDC et Event
C1D3  CDA0CC   CALL   0CCA0H      ;détourner vecteurs cassette
C1D6  E1       POP    HL          ;transmettre nouvelles valeurs pour
C1D7  D1       POP    DE          ;lomem et himem à KL START PROGRAM
```

```
C1D8  FDE1      POP    IY
C1DA  37        SCF                 ;marque initialisation OK
C1DB  C9        RET

*************** ENTRY démarrage à froid CP/M
C1DC  3100C0    LD     SP,0C000H    ;initialiser pile
C1DF  FD2148AC  LD     IY,0AC48H
C1E3  1133AD    LD     DE,0AD33H
C1E6  01A500    LD     BC,00A5H
C1E9  CDAFCA    CALL   0CAAFH       ;efface (de) à (de+bc)
C1EC  2141AD    LD     HL,0AD41H
C1EF  35        DEC    (HL)
C1F0  3E81      LD     A,81H        ;fixation standard octet IO
C1F2  320300    LD     (0003H),A
C1F5  AF        XOR    A            ;Drive et User
C1F6  320400    LD     (0004H),A
C1F9  2133C0    LD     HL,0C033H    ;table jump instructions Controller
                                    &81-&89
C1FC  1180BE    LD     DE,0BE80H    ;copier dans BE80
C1FF  013F00    LD     BC,003FH     ;en tout 3fh octets
C202  EDB0      LDIR                ;transmettre
C204  CDC0C0    CALL   0C0C0H       ;sauver vecteur INT & adresse port GA
C207  CDDDC5    CALL   0C5DDH       ;initialiser FDC et Event
C20A  0E41      LD     C,41H        ;numéro de secteur
C20C  110000    LD     DE,0000H     ;piste et lecteur
C20F  210001    LD     HL,0100H     ;adresse buffer
C212  CD66C6    CALL   0C666H       ;lire secteur
C215  DCACC2    CALL   C,0C2ACH     ;=> secteur lu vide?
C218  300A      JR     NC,0C224H    ;erreur apparue
C21A  EB        EX     DE,HL
C21B  017FC1    LD     BC,0C17FH
C21E  3133AD    LD     SP,0AD33H
C221  C377C1    JP     0C177H       ;vecteur INT sur C163, JP (DE)

*************** erreur lors du chargement du secteur BOOT
C224  3E0F      LD     A,0FH        ;Msg. 15 'Failed to load Boot sector'
C226  CDB8CA    CALL   0CAB8H       ;demander 'CHAN., IGN. or RETRY'
C229  18DF      JR     0C20AH

*************** charger CP/M CCP et BDOS à partir disque, Warm Boot
C22B  CD6FC8    CALL   0C86FH
C22E  CDB0C8    CALL   0C8B0H
```

```
C231  014801   LD     BC,0148H      ;b=compteur secteur, c=numéro secteur
C234  110000   LD     DE,0000H      ;piste et lecteur
C237  E5       PUSH   HL            ;ranger adresse buffer
C238  CD99C2   CALL   0C299H        ;lire nombre secteurs voulu
C23B  E1       POP    HL            ;début buffer à nouveau dans hl
C23C  DCACC2   CALL   C,0C2ACH      ;teste, si secteur lu est vide
C23F  3051     JR     NC,0C292H     ;erreur apparue
C241  E5       PUSH   HL
C242  23       INC    HL
C243  5E       LD     E,(HL)
C244  23       INC    HL
C245  56       LD     D,(HL)
C246  21A4FC   LD     HL,0FCA4H
C249  19       ADD    HL,DE
C24A  EB       EX     DE,HL
C24B  E1       POP    HL
C24C  010002   LD     BC,0200H      ;longueur buffer
C24F  EDB0     LDIR
C251  EB       EX     DE,HL         ;adresse buffer
C252  01490A   LD     BC,0A49H      ;b=compteur secteur, c=numéro secteur
C255  110000   LD     DE,0000H      ;piste et lecteur
C258  CD99C2   CALL   0C299H        ;lire nombre secteurs voulu
C25B  3035     JR     NC,0C292H     ;erreur apparue
C25D  EB       EX     DE,HL
C25E  2100EA   LD     HL,0EA00H
C261  19       ADD    HL,DE
C262  E5       PUSH   HL
C263  2106F2   LD     HL,0F206H
C266  19       ADD    HL,DE
C267  3EC3     LD     A,0C3H        ;code d'opération pour JP
C269  320500   LD     (0005H),A     ;
C26C  220600   LD     (0006H),HL    ;Entry BDOS, 8F00
C26F  320000   LD     (0000H),A     ;Code JP
C272  210300   LD     HL,0003H
C275  19       ADD    HL,DE
C276  220100   LD     (0001H),HL    ;Entry BIOS, AD00
C279  217FC1   LD     HL,0C17FH     ;Table des vecteurs CP/M
C27C  013300   LD     BC,0033H      ;51 octets de long
C27F  EDB0     LDIR                 ;dans AD00
C281  210400   LD     HL,0004H      ;User et lecteur
C284  7E       LD     A,(HL)
C285  E60F     AND    0FH           ;isoler lecteur
```

```
C287  FE02     CP     02H          ;lecteur 0 ou 1?
C289  3802     JR     C,0C28DH     ;est 0 ou 1 =>
C28B  3600     LD     (HL),00H     ;sinon mettre sur 0
C28D  4E       LD     C,(HL)
C28E  D1       POP    DE
C28F  C377C1   JP     0C177H       ;vecteur INT sur C163, JP(de)

*************** erreur apparue lors du chargement de CP/M
C292  3E0E     LD     A,0EH        ;Msg. 14 'Failed to load CPM'
C294  CDB8CA   CALL   0CAB8H       ;demander 'CHAN., IGN. or RETRY'
C297  1892     JR     0C22BH       ;WARM BOOT

** charge continuellement secteurs, nombre en b, secteur en c, piste en d
C299  CD66C6   CALL   0C666H       ;charger secteur
C29C  D0       RET    NC           ;erreur apparue
C29D  79       LD     A,C          ;numéro secteur dans accu
C29E  0C       INC    C            ;prochain secteur
C29F  FE49     CP     49H          ;dernier secteur était Nr 49h?
C2A1  3803     JR     C,0C2A6H     ;=> n'était pas 49h
C2A3  0E41     LD     C,41H        ;Sector 41h
C2A5  14       INC    D            ;sur piste suivante
C2A6  24       INC    H            ;élever pointeur buffer de 2 pages
C2A7  24       INC    H
C2A8  10EF     DJNZ   0C299H       ;lu tous les secteurs?
C2AA  37       SCF                 ;marque que tout est OK
C2AB  C9       RET

*************** teste si secteur lu est vide
C2AC  E5       PUSH   HL           ;adresse buffer
C2AD  010200   LD     BC,0002H     ;512 octets
C2B0  7E       LD     A,(HL)       ;premier caract. de buffer dans accu
C2B1  BE       CP     (HL)         ;(pointeur buffer)
C2B2  23       INC    HL           ;augmenter pointeur
C2B3  37       SCF                 ;marque OK
C2B4  2006     JR     NZ,0C2BCH    ;si différent, alors =>
C2B6  10F9     DJNZ   0C2B1H       ;boucle sur 256 octets
C2B8  0D       DEC    C            ;fois deux = un secteur
C2B9  20F6     JR     NZ,0C2B1H
C2BB  B7       OR     A            ;annuler Carry
C2BC  E1       POP    HL           ;répéter pointeur buffer
C2BD  C9       RET
```

```
*************** WARM BOOT
C2BE  CD33AD    CALL    0AD33H      ;'JP C22B'
C2C1  2BC2      DEFW    0C22BH

*************** CONSOLE INPUT
C2C3  2186C4    LD      HL,0C486H   ;(hl)=> Affectation Console In
C2C6  181C      JR      0C2E4H

*************** CONSOLE OUTPUT
C2C8  218FC4    LD      HL,0C48FH   ;(hl)=> Affectation Console Out
C2CB  1817      JR      0C2E4H

*************** PRINTER STATUS
C2CD  2198C4    LD      HL,0C498H   ;(hl)=> Affect. List Device Status
C2D0  1812      JR      0C2E4H

*************** PRINTER OUTPUT
C2D2  21A1C4    LD      HL,0C4A1H   ;(hl)=> Affect. List Device Output
C2D5  180D      JR      0C2E4H

*************** PUNCHER
C2D7  21AAC4    LD      HL,0C4AAH   ;(hl)=> Affectation Puncher
C2DA  1808      JR      0C2E4H

*************** READER
C2DC  21BCC4    LD      HL,0C4BCH   ;(hl)=> Affectation Reader
C2DF  1803      JR      0C2E4H

*************** CONSOLE STATUS
C2E1  217DC4    LD      HL,0C47DH   ;(hl)=> Affectation Console Status
C2E4  CD33AD    CALL    0AD33H      ;'JP C46A'; Affect. par octet I/O
C2E7  6AC4

*************** CHERCHER PISTE 0
C2E9  CD68C1    CALL    0C168H      ;'JP C51F'
C2EC  1FC5
C2EE  2189BE    LD      HL,0BE89H
C2F1  C9        RET

*************** SELECT DRIVE
C2F2  CD33AD    CALL    0AD33H      ;'JP C4F0'
C2F5  F0C4
```

```
*************** READ SECTOR
C2F7  CD33AD   CALL    0AD33H      ;'JP C54C'
C2FA  4CC5

*************** WRITE SECTOR
C2FC  CD33AD   CALL    0AD33H      ;'JP C52E'
C2FF  2EC5

***************
C301  32C5AD   LD      (0ADC5H),A
C304  018100   LD      BC,0081H
C307  1142AD   LD      DE,0AD42H
C30A  EDB0     LDIR
C30C  2143AD   LD      HL,0AD43H
C30F  22C3AD   LD      (0ADC3H),HL
C312  C9       RET

*************** tester état clavier, caractère disponible?
C313  2141AD   LD      HL,0AD41H
C316  7E       LD      A,(HL)
C317  B7       OR      A
C318  2804     JR      Z,0C31EH
C31A  35       DEC     (HL)
C31B  CC81BB   CALL    Z,0BB81H    ;TXT CURSOR ON
C31E  CD09BB   CALL    0BB09H      ;KM READ CHAR, 1 caract. du clavier
C321  DC0CBB   CALL    C,0BB0CH    ;si car. disponible, KM RETURN CHAR
C324  9F       SBC     A,A         ;0ffh=> caractère disponible,
C325  C9       RET                 ;00 pas de caractère
```

```
*************** Console Input, retirer un caractère du clavier
C326  2142AD    LD      HL,0AD42H   ;Keyboard Modus Flag
C329  7E        LD      A,(HL)
C32A  B7        OR      A           ;tester flag
C32B  281B      JR      Z,0C348H    ;=> Keyboard Mode 'INPUT'
C32D  CD09BB    CALL    0BB09H      ;Keyboard Mode 'INKEY', KM READ CHAR
C330  300C      JR      NC,0C33EH   ;=> reçu aucun caractère
C332  21C5AD    LD      HL,0ADC5H
C335  34        INC     (HL)
C336  35        DEC     (HL)
C337  C0        RET     NZ
C338  2142AD    LD      HL,0AD42H
C33B  3600      LD      (HL),00H
C33D  C9        RET

***************
C33E  35        DEC     (HL)        ;Keyboard Mode sur 'INPUT'
C33F  2AC3AD    LD      HL,(0ADC3H)
C342  7E        LD      A,(HL)
C343  23        INC     H
L
C344  22C3AD    LD      (0ADC3H),HL
C347  C9        RET

*************** Console Input, attendre un caractère du clavier
C348  2141AD    LD      HL,0AD41H
C34B  7E        LD      A,(HL)
C34C  B7        OR      A
C34D  C481BB    CALL    NZ,0BB81H   ;TXT CURSOR ON
C350  3600      LD      (HL),00H
C352  C306BB    JP      0BB06H      ;KM WAIT CHAR, attendre caractère

*************** High Speed Reader comme Reader, non étendu
C355  3E1A      LD      A,1AH       ;EOF
C357  C9        RET

*************** Status CRT comme Printer, High Speed Reader comme Reader
C358  3EFF      LD      A,0FFH

*************** High Speed Puncher comme Puncher Device
C35A  C9        RET
```

```
*************** CRT-Device, sortir un caractère sur l'écran
C35B  2141AD   LD     HL,0AD41H
C35E  7E       LD     A,(HL)
C35F  B7       OR     A
C360  CC84BB   CALL   Z,0BB84H    ;TXT CURSOR OFF
C363  36FF     LD     (HL),0FFH
C365  79       LD     A,C         ;caractère à sortir dans l'accu
C366  CD5ABB   CALL   0BB5AH      ;TXT OUTPUT
C369  FE20     CP     20H         ;caractère espace
C36B  D0       RET    NC          ;=> pas code de contrôle
C36C  CD78BB   CALL   0BB78H      ;TXT GET CURSOR
C36F  CD87BB   CALL   0BB87H      ;TXT VALIDATE
C372  D8       RET    C
C373  CD8ABB   CALL   0BB8AH      ;TXT PLACE CURSOR
C376  C38DBB   JP     0BB8DH      ;TXT REMOVE CURSOR

*************** Line Printer Status, teste, si Centronics Busy
C379  CD2EBD   CALL   0BD2EH      ;MC BUSY PRINTER, Carry si Busy
C37C  3F       CCF                ;Carry, si pas Busy
C37D  9F       SBC    A,A         ;0ffh => pas Busy, 00 si pas Busy
C37E  C9       RET

*************** Line Printer Output, un caractère vers l'imprimante
C37F  79       LD     A,C         ;caractère dans accu
C380  CD2BBD   CALL   0BD2BH      ;MC PRINT CHAR, sortir
C383  D8       RET    C           ;envoi caractère réussi
C384  CDD3C4   CALL   0C4D3H      ;Printer Busy, tester Keyboard
C387  18F6     JR     0C37FH      ;nouvelle tentative

*************** initialiser I/O sérieI  (hl)=> table paramètres
C389  F3       DI
C38A  01DDFA   LD     BC,0FADDH   ;SIO Canal A/registre de contrôle
C38D  11C6AD   LD     DE,0ADC6H   ;mémoire pour registre WR 5, Canal A
C390  CDBDC3   CALL   0C3BDH      ;Reset canal, Init. Can. A
C393  03       INC    BC
C394  03       INC    BC          ;SIO Canal B/registre de contrôle
C395  13       INC    DE          ;mémoire pour registre WR 5, Canal B
C396  CDBDC3   CALL   0C3BDH      ;Channel Reset, Init. Can. B
C399  3E36     LD     A,36H       ;Mode Timer 0 du 8253
C39B  1EDC     LD     E,0DCH      ;octet faible pour adr. port timer 0
C39D  CDAEC3   CALL   0C3AEH      ;fixer baudrate d'envoi canal A
C3A0  3E76     LD     A,76H       ;Mode Timer 1 du 8253
```

```
C3A2  1C        INC     E           ;octet faible pour adr. port Timer 1
C3A3  CDAEC3    CALL    0C3AEH      ;fixer baud rate réception canal A
C3A6  3EB6      LD      A,0B6H      ;Mode Timer 2 du 8253
C3A8  1C        INC     E           ;octet faible pour adr. port Timer 2
C3A9  CDAEC3    CALL    0C3AEH      ;Baudrate réc. et envoi canal B
C3AC  FB        EI
C3AD  C9        RET

************** init. Baudrate-Generator 8253, (hl) => Lo-Hi valeurs timer
C3AE  01DFFB    LD      BC,0FBDFH   ;adr. port mot contrôle 8253
C3B1  ED79      OUT     (C),A       ;sortir mot de contrôle au 8253
C3B3  4B        LD      C,E         ;octet faible adr.port du timer voulu
C3B4  7E        LD      A,(HL)      ;octet faible valeur timer
C3B5  23        INC     HL
C3B6  ED79      OUT     (C),A       ;charger dans timer
C3B8  7E        LD      A,(HL)      ;octet fort valeur timer
C3B9  23        INC     HL
C3BA  ED79      OUT     (C),A       ;charger dans timer
C3BC  C9        RET

*************** initialiser canal SIO dans (BC)
C3BD  3E18      LD      A,18H       ; code d'opération restaurer canal
C3BF  ED79      OUT     (C),A       ;sortir sur SIO
C3C1  3E04      LD      A,04H       ;sélectionner Write-Register 4
C3C3  ED79      OUT     (C),A
C3C5  7E        LD      A,(HL)      ;entrée de table pour
C3C6  23        INC     HL          ;Parity, Stop-Bits et Clock-Mode
C3C7  ED79      OUT     (C),A       ;sortir sur SIO
C3C9  3E05      LD      A,05H       ;sélectionner Write-Register 5
C3CB  ED79      OUT     (C),A
C3CD  7E        LD      A,(HL)      ;ranger entrée table pour hand shake
C3CE  12        LD      (DE),A      ;et Bits/Char dans (de) (ADC6/ADC7)
C3CF  23        INC     HL
C3D0  ED79      OUT     (C),A       ;et sortir sur SIO, envoi param. (TX)
C3D2  3E03      LD      A,03H       ;sélectionner Write-Register 3
C3D4  ED79      OUT     (C),A
C3D6  7E        LD      A,(HL)      ;valeur table pour Handshake et
C3D7  23        INC     HL          ;Bits/Char.
C3D8  ED79      OUT     (C),A       ;sortir sur SIO, param. récept. RX
C3DA  C9        RET

*************** Canal A buffer RX plein?
```

```
C3DB  01DDFA    LD      BC,0FADDH   ;SIO Canal A, registre de contrôle
C3DE  21C6AD    LD      HL,0ADC6H   ;(hl)=> contenu Write-Reg. 5,Can. A
C3E1  1806      JR      0C3E9H

*************** Canal B buffer RX plein?
C3E3  01DFFA    LD      BC,0FADFH   ;SIO Canal B, registre de contrôle
C3E6  21C7AD    LD      HL,0ADC7H   ;(hl)=> Write-Reg. 5, Can. B
C3E9  ED78      IN      A,(C)       ;lire Read-Reg. 0 du canal voulu
C3EB  0F        RRCA                ;Bit 0, caractère RX prêt?
C3EC  9F        SBC     A,A
C3ED  D8        RET     C           ;=> un caractère présent
C3EE  CD24C4    CALL    0C424H      ;fixer bit DTR
C3F1  ED78      IN      A,(C)       ;lire Read-Reg. 0
C3F3  0F        RRCA                ;Bit 0, caractère RX prêt?
C3F4  9F        SBC     A,A
C3F5  1829      JR      0C420H      ;annuler bit DTR

*************** SIO canal A retirer un caractère
C3F7  01DDFA    LD      BC,0FADDH   ;SIO canal A, registre de contrôle
C3FA  21C6AD    LD      HL,0ADC6H   ;(hl)=> contenu Write-Reg. 5, Can. A
C3FD  1806      JR      0C405H

*************** SIO canal B retirer un caractère
C3FF  01DFFA    LD      BC,0FADFH   ;SIO canal B, registre de contrôle
C402  21C7AD    LD      HL,0ADC7H   ;(hl)=> Write-Reg. 5, Can. B
C405  ED78      IN      A,(C)       ;lire Read-Reg. 0
C407  0F        RRCA                ;caractère RX disponible?
C408  3812      JR      C,0C41CH    ;=> recevoir un caractère
C40A  CD24C4    CALL    0C424H      ;fixer bit DTR
C40D  CDC5C4    CALL    0C4C5H      ;interroger clavier
C410  FE1A      CP      1AH         ;entré Control Z comme fin?
C412  280C      JR      Z,0C420H    ;=> annuler bit DTR, RET
C414  ED78      IN      A,(C)       ;lire Read-Reg. 0
C416  0F        RRCA                ;Caractère RX disponible?
C417  30F4      JR      NC,0C40DH   ;=> pas encore disponible
C419  CD20C4    CALL    0C420H      ;annuler bit DTR
C41C  0B        DEC     BC          ;adresse de port registre de données
C41D  ED78      IN      A,(C)       ;lire caractère reçu
C41F  C9        RET
```

```
*************** annuler bit DTR, autoriser réception
C420  1E00      LD     E,00H       ;annuler bit 7, indifférent
C422  1802      JR     0C426H

*************** mettre bit DTR, interdire réception
C424  1E80      LD     E,80H       ;mettre bit 7
C426  F3        DI
C427  F5        PUSH   AF
C428  3E05      LD     A,05H       ;Write-Reg. 5
C42A  ED79      OUT    (C),A       ;appeler
C42C  7E        LD     A,(HL)      ;(hl)=> contenu de WR-Reg 5
C42D  E67F      AND    7FH         ;isoler bit 7
C42F  B3        OR     E           ;suivant appel Bit 7 mis/annulé
C430  ED79      OUT    (C),A       ;écrire dans Write-Reg. 5
C432  F1        POP    AF
C433  FB        EI
C434  C9        RET

*************** teste, si buffer TX canal A vide
C435  01DDFA    LD     BC,0FADDH   ;SIO canal A, registre de contrôle
C438  1803      JR     0C43DH

                              *************** teste, si buffer TX canal B vide
C43A  01DFFA    LD     BC,0FADFH   ;SIO canal B, registre de contrôle
C43D  ED78      IN     A,(C)       ;lire Read-Reg 0
C43F  E604      AND    04H         ;isoler TX-Empty-Bit
C441  C8        RET    Z           ;=> Buffer n'est pas vide
C442  37        SCF                ;marque buffer vide
C443  9F        SBC    A,A
C444  C9        RET

*************** envoyer un caractère à travers canal A
C445  79        LD     A,C         ;caractère à sortir dans accu
C446  01DDFA    LD     BC,0FADDH   ;SIO canal A, registre de contrôle
C449  1804      JR     0C44FH

*************** envoyer un caractère à travers canal B
C44B  79        LD     A,C         ;caractère à sortir dans accu
C44C  01DFFA    LD     BC,0FADFH   ;SIO canal B, registre de contrôle
C44F  F5        PUSH   AF          ;sauver caractère
C450  CDD3C4    CALL   0C4D3H      ;interroger clavier
C453  CD3DC4    CALL   0C43DH      ;buffer d'envoi vide?
```

```
C456  30F8      JR     NC,0C450H   ;=> pas encore vide
C458  F1        POP    AF          ;caractère à nouveau dans accu
C459  0B        DEC    BC          ;adresse de port registre de données
C45A  ED79      OUT    (C),A       ;écrire caractère dans SIO
C45C  C9        RET

*************** déterminer READER Status à travers octet I/O
C45D  21B3C4    LD     HL,0C4B3H   ;table READER-Status
C460  1808      JR     0C46AH

*************** READER-Input à travers octet I/O
C462  21BCC4    LD     HL,0C4BCH   ;table READER-Input
C465  1803      JR     0C46AH

*************** PRINTER-OUTPUT à travers octet I/O
C467  21A1C4    LD     HL,0C4A1H   ;table PRINTER-Output

******** déterminer I/O-Device avec octet I/O, (hl)=> table d'affectation
C46A  46        LD     B,(HL)      ;nombre boucle pour les 4 Devices
C46B  23        INC    HL          ;(hl)=> première affectation
C46C  3A0300    LD     A,(0003H)   ;octet I/O, habituellement &81
C46F  07        RLCA
C470  10FD      DJNZ   0C46FH      ;(b) fois octet I/O vers la gauche
C472  E606      AND    06H         ;isoler bits significatifs
C474  1600      LD     D,00H
C476  5F        LD     E,A         ;donne décalage dns table affectation
C477  19        ADD    HL,DE       ;additionner à start
C478  5E        LD     E,(HL)      ;Adresse de routine I/O dans de
C479  23        INC    HL
C47A  56        LD     D,(HL)
C47B  EB        EX     DE,HL
C47C  E9        JP     (HL)        ;saut indirect à routine I/O

*************** CONSOLE STATUS
C47D  01        DEFB   01H
C47E  A7BE      DEFW   0BEA7H      ;JP 0C3D8H,Car.SIO Can. A disponible?
C480  13C3      DEFW   0C313H      ;Caractère du clavier disponible?
C482  5DC4      DEFW   0C45DH      ;READER Status à travers octet I/O
C484  B3BE      DEFW   0BEB3H      ;JP 0C3E3H,Car.SIO Can. B disponible?

*************** CONSOLE INPUT
C486  01        DEFB   01H
```

```
C487  AABE       DEFW   0BEAAH      ;JP 0C3F7H,retirer car. de SIO Can. A
C489  26C3       DEFW   0C326H      ;retirer caractère du clavier
C48B  62C4       DEFW   0C462H      ;lire car. READER à travers octet I/O
C48D  B6BE       DEFW   0BEB6H      ;JP 0C3FFH,retirer car. de SIO Can. B

*************** CONSOLE OUTPUT
C48F  01         DEFB   01H
C490  B0BE       DEFW   0BEB0H      ;JP 0C445H,envoyer car. par SIO Can.A
C492  5BC3       DEFW   0C35BH      ;sortir caractère sur l'écran
C494  67C4       DEFW   0C467H      ;PRINTER OUTPUT à travers octet I/O
C496  BCBE       DEFW   0BEBCH      ;JP 0C44BH,envoyer car. par SIO Can.B

*************** PRINTER STATUS
C498  03         DEFB   03H
C499  ADBE       DEFW   0BEADH      ;JP 0C435, buffer envoi can. A vide?
C49B  58C3       DEFW   0C358H      ;non complété
C49D  79C3       DEFW   0C379H      ;teste Centronics Busy
C49F  B9BE       DEFW   0BEB9H      ;JP 0C43A, buffer envoi can. B vide?

*************** PRINTER OUTPUT
C4A1  03         DEFB   03H
C4A2  B0BE       DEFW   0BEB0H      ;JP 0C445H,envoyer car. par SIO Can.A
C4A4  5BC3       DEFW   0C35BH      ;sortir caractère sur l'écran
C4A6  7FC3       DEFW   0C37FH      ;sort caractère sur port Centronics
C4A8  BCBE       DEFW   0BEBCH      ;JP 0C44BH,envoyer car. par SIO Can.B

*************** PUNCHER
C4AA  05         DEFB   05H
C4AB  B0BE       DEFW   0BEB0H      ;JP 0C445H,envoyer car. par SIO Can.A
C4AD  5AC3       DEFW   0C35AH      ;non complété, RET
C4AF  BCBE       DEFW   0BEBCH      ;JP 0C44BH,envoyer car. par SIO Can.B
C4B1  5BC3       DEFW   0C35BH      ;sortir caractère sur l'écran

*************** READER Status
C4B3  07         DEFB   07H
C4B4  A7BE       DEFW   0BEA7H      ;JP 0C3D8H, Car. SIO Can. A dispon.?
C4B6  58C3       DEFW   0C358H      ;non complété
C4B8  B3BE       DEFW   0BEB3H      ;JP 0C3E3H, Car. SIO Can. B? dispon.?
C4BA  13C3       DEFW   0C313H      ;teste, si car. du clavier présent

*************** READER lire caractère
C4BC  07         DEFB   07H
```

```
C4BD  AABE      DEFW   OBEAAH     ;JP OC3F7H, retire car. de SIO can.A
C4BF  55C3      DEFW   OC355H     ;non complété, retire EOF
C4C1  B6BE      DEFW   OBEB6H     ;JP OC3FFH, retire car. de SIO can.B
C4C3  26C3      DEFW   OC326H     ;retire caractère du clavier

***************
C4C5  CDD3C4    CALL   OC4D3H     ;tester si CONTROL C enfoncé
C4C8  FE13      CP     13H        ;ENTER ?
C4CA  CO        RET    NZ         ;=> pas ENTER
C4CB  E5        PUSH   HL
C4CC  C5        PUSH   BC
C4CD  CD26C3    CALL   OC326H     ;retirer autre caractère du clavier
C4DO  C1        POP    BC
C4D1  E1        POP    HL
C4D2  C9        RET

*************** tester, si CONTROL C est appuyé
C4D3  E5        PUSH   HL
C4D4  D5        PUSH   DE
C4D5  C5        PUSH   BC
C4D6  CD13C3    CALL   OC313H     ;tester Keyboard Status
C4D9  B7        OR     A
C4DA  280F      JR     Z,OC4EBH   ;=> pas de caractère disponible
C4DC  CD26C3    CALL   OC326H     ;retirer caractère du clavier
C4DF  FE03      CP     03H        ;CONTROL C?
C4E1  2008      JR     NZ,OC4EBH  ;=> pas Control C
C4E3  3EOD      LD     A,ODH      ;message système 14 ..^C
C4E5  CDEBCA    CALL   OCAEBH     ;sortir
C4E8  C32BC2    JP     OC22BH     ;Warm Boot

*************** pas d'action, CONTROL C pas appuyé
C4EB  C1        POP    BC
C4EC  D1        POP    DE
C4ED  E1        POP    HL
C4EE  C9        RET

*************** pas utilisé
C4EF  FF        RST    38H

*************** SELECT DRIVE
C4FO  79        LD     A,C        ;No lecteur dans accu
C4F1  FE02      CP     02H        ;ne peut être que 0 ou 1
```

```
C4F3  210000     LD      HL,0000H
C4F6  D0         RET     NC            ;No lecteur trop grand
C4F7  7B         LD      A,E           ;No lecteur actuel dans accu
C4F8  1F         RRA                   ;Bit 0 dans Carry
C4F9  380F       JR      C,0C50AH      ;Saut, si lecteur 1 actif jusqu'ici
C4FB  59         LD      E,C           ;lecteur voulu dans e
C4FC  3E18       LD      A,18H
C4FE  CD5CCA     CALL    0CA5CH        ;charge val. bloc param.disc 18h dns
                                        accu
C501  B7         OR      A             ;si différent 0
C502  2006       JR      NZ,0C50AH     ;alors saut
C504  E5         PUSH    HL
C505  CD6CC5     CALL    0C56CH        ;sinon déterminer format disque
C508  E1         POP     HL
C509  D0         RET     NC
C50A  79         LD      A,C           ;numéro lecteur dans accu
C50B  3253BE     LD      (0BE53H),A    ;buffer HS/US
C50E  211002     LD      HL,0210H      ;décal. Disc Param. Header lecteur A
C511  B7         OR      A             ;si lecteur A utilisé, alors saut,
C512  2803       JR      Z,0C517H
C514  212002     LD      HL,0220H      ;sinon décalage DPH pour lecteur B
C517  C39FCA     JP      0CA9FH        ;hl=hl+iy, hl => début table pointeur

*************** entrer adresse buffer enregistrement
C51A  ED4360BE   LD      (0BE60H),BC ;(bc):= buffer enregistrement
C51E  C9         RET

*************** chercher piste 0
C51F  CD6FC8     CALL    0C86FH
C522  0E00       LD      C,00H
C524  79         LD      A,C
C525  3254BE     LD      (0BE54H),A  ;(Buffer pour numéro de piste)
C528  C9         RET

*************** envoyer numéro enregistrement au controller
C529  79         LD      A,C
C52A  3255BE     LD      (0BE55H),A  ;(Buffer pour numéro enregistrement)
C52D  C9         RET

*************** Write Record (écrire enregistrement)
C52E  C5         PUSH    BC
C52F  79         LD      A,C
```

```
C530  FE02      CP      02H
C532  CCEBC7    CALL    Z,0C7EBH     ;lecteur, piste, enr. de be53h dns
                                      be5ah
C535  CD00C8    CALL    0C800H       ;teste, si Drv, Trk & Rec dns be53h =
                                      be5ah
C538  DC1BC8    CALL    C,0C81BH     ;=> identiques
C53B  CD32C8    CALL    0C832H       ;déterminer secteur avec numéro enr.
C53E  C1        POP     BC
C53F  D0        RET     NC
C540  CDB6C8    CALL    0C8B6H       ;transfère enreg. dans buffer secteur
C543  0D        DEC     C            ;nombre enregistrements
C544  37        SCF
C545  CC6FC8    CALL    Z,0C86FH     ;ajouter décal.secteur,écrire secteur
C548  D0        RET     NC           ;=> erreur
C549  3E00      LD      A,00H
C54B  C9        RET

*************** Read Record
C54C  AF        XOR     A            ;vider accu et
C54D  3259BE    LD      (0BE59H),A   ;entrer dans Blockmask+1
C550  CD32C8    CALL    0C832H       ;déterminer secteur avec No enreg.
C553  CDC7C8    CALL    0C8C7H       ;transfère enreg. dans buffer enreg.
C556  D0        RET     NC
C557  3E00      LD      A,00H
C559  C9        RET

*************** traduire numéro enregistrement
C55A  60        LD      H,B          ;transfère seulement bc dans hl
C55B  69        LD      L,C
C55C  C9        RET

*************** lire ID secteur, interroger erreurs éventuelles
C55D  017EFB    LD      BC,0FB7EH    ;Status reg. FDC
C560  3E4A      LD      A,4AH        ;lire code ID secteur
C562  CD5CC9    CALL    0C95CH       ;sortir accu sur FDC
C565  7B        LD      A,E          ;Unit Select/Head Select
C566  CD5CC9    CALL    0C95CH       ;sortir accu sur FDC
C569  C3F9C8    JP      0C8F9H       ;phase résultat FDC, Drive READY?

*************** déterminer format disquette d'après ID secteur
C56C  CD76C9    CALL    0C976H       ;Moteur en marche
C56F  3E16      LD      A,16H
```

```
C571  CD5CCA    CALL    0CA5CH      ;charger  valeur  bloc  paramètres
                                     disque 16h dans accu
C574  57        LD      D,A         ;c'est le numéro de piste actuel
C575  0E10      LD      C,10H       ;nombre de tentatives de lecture
C577  215DC5    LD      HL,0C55DH   ;Adresse Routine 'lire ID secteur'
C57A  CDFFC6    CALL    0C6FFH      ;chercher piste dans d
C57D  D0        RET     NC          ;NC = erreur
C57E  3A51BE    LD      A,(0BE51H)  ;No secteur de FDC dns phase résultat

*************** ^83h déterminer formatage disque
C581  F5        PUSH    AF          ;contient numéro secteur lu par FDC
C582  AF        XOR     A           ;vider accu
C583  CD63CA    CALL    0CA63H      ;début bloc param. disque dans hl
C586  E5        PUSH    HL
C587  EB        EX      DE,HL
C588  2143CA    LD      HL,0CA43H   ;bloc param.disc stndrd (DPB) dns Rom
C58B  011600    LD      BC,0016H    ;22 octets
C58E  EDB0      LDIR                ;dans bloc param. disque actuel
C590  E1        POP     HL          ;début table
C591  F1        POP     AF          ;numéro de secteur
C592  E6C0      AND     0C0H        ;annuler bits 0 à 5
C594  FE40      CP      40H         ;bit 6 mis?
C596  37        SCF
C597  C8        RET     Z           ;alors utiliser table format standard
C598  11CAC5    LD      DE,0C5CAH   ;début format données DPB, Tab.2
C59B  FEC0      CP      0C0H        ;Bit 6 et 7 mis?
C59D  2803      JR      Z,0C5A2H    ;si oui, utiliser format de données
C59F  11C0C5    LD      DE,0C5C0H   ;début DBP format IBM, Tab.1
C5A2  1A        LD      A,(DE)      ;transférer les deux 1ères valeurs
C5A3  13        INC     DE          ;dans bloc param. disc
C5A4  77        LD      (HL),A
C5A5  23        INC     HL
C5A6  1A        LD      A,(DE)
C5A7  13        INC     DE
C5A8  77        LD      (HL),A
C5A9  010400    LD      BC,0004H    ;
C5AC  09        ADD     HL,BC       ;hl pointe sur bloc param. disc +5
C5AD  1A        LD      A,(DE)      ;nombre blocs/disque
C5AE  13        INC     DE          ;transférer deux octets suivants
C5AF  77        LD      (HL),A      ;dans bloc param. disque
C5B0  23        INC     HL
C5B1  1A        LD      A,(DE)
```

```
C5B2  13        INC   DE
C5B3  77        LD    (HL),A
C5B4  010700    LD    BC,0007H
C5B7  09        ADD   HL,BC      ;hl pointe sur bloc param. disque +13
C5B8  EB        EX    DE,HL
C5B9  010600    LD    BC,0006H   ;transférer les 6 octets restants
C5BC  EDB0      LDIR             ;dans bloc paramètres disque
```

```
C5BE  37          SCF
C5BF  C9          RET

*************** valeurs table pour format IBM
C5C0  2000        DEFW    0020        ;nombre enregistr./piste, SPT
C5C2  9B00        DEFW    009B        ;nombre blocs/disque, DSM
C5C4  0100        DEFW    0001        ;nombre pistes pour système
                                       d'exploitation, OFF
C5C6  01          DEFB    01          ;décalage secteur
C5C7  08          DEFB    08          ;secteurs/piste
C5C8  2A          DEFB    2A          ;longueur GAP 3 read/write
C5C9  50          DEFB    50          ;longueur GAP 3 formater

*************** valeurs table pour format disque de données
C5CA  2400        DEFW    0024        ;Nombre enreg./piste, SPT
C5CC  B300        DEFW    00B3        ;Nombre blocs/disque, DSM
C5CE  0000        DEFW    0000        ;Nombre pour système d'exploitation,
                                       OFF
C5D0  C1          DEFB    C1          ;décalage secteur
C5D1  09          DEFB    09          ;secteurs/piste
C5D2  2A          DEFB    2A          ;longueur GAP 3 read/write
C5D3  52          DEFB    52          ;longueur GAP 3 formater

*************** table (7 octets) est copiée dans be44...
C5D4  3200        DEFW    0032        ;délai plein régime moteur disque
C5D6  FA00        DEFW    00FA        ;ticker délai pour moteur lecteur
C5D8  AF          DEFB    AF          ;
C5D9  0F0C        DEFW    0C0F        ;valeurs pour longue boucle délai

*************** Table (2 octets) nécessaire en &82
C5DB  01          DEFB    01          ;Head Unload Time pour FDC = 32 ms
C5DC  03          DEFB    03          ;Head Load Time pour FDC = 16 ms
*************** initialise DPHs, DPBs, FDC et Event
C5DD  1140BE      LD      DE,0BE40H   ;Adresse Ram pour routines controller
C5E0  013D00      LD      BC,003DH    ;Nombre d'octets
C5E3  CDAFCA      CALL    0CAAFH      ;efface (de) à (de+bc)
C5E6  CDF4C9      CALL    0C9F4H      ;initialise bloc paramètres
                                       disque/Header
C5E9  CDE8C9      CALL    0C9E8H      ;arrêter moteur lecteur
C5EC  21D4C5      LD      HL,0C5D4H   ;table paramètres FDC
C5EF  CD0DC6      CALL    0C60DH      ;initialiser paramètres lecteur FDC
C5F2  CD12B9      CALL    0B912H      ;KL ASK CURR SELECTION
```

```
C5F5  4F        LD      C,A            ;sélection Rom pour routine Event
C5F6  0680      LD      B,80H          ;classe Event asynchrone
C5F8  216DBE    LD      HL,0BE6DH      ;bloc Event
C5FB  11D6C9    LD      DE,0C9D6H      ;Adresse de la routine Event
C5FE  CDEFBC    CALL    0BCEFH         ;KL INIT EVENT
C601  3E10      LD      A,10H

*************** ↑89 fixe nombre tentatives lecture, nombre dans accu
C603  E5        PUSH    HL
C604  2A66BE    LD      HL,(0BE66H) ;Nombre tentatives lecture
C607  3266BE    LD      (0BE66H),A
C60A  7D        LD      A,L
C60B  E1        POP     HL
C60C  C9        RET

*************** ↑82 spécifier données lecteur
C60D  1144BE    LD      DE,0BE44H      ;FDC-Ram +4
C610  010700    LD      BC,0007H
C613  EDB0      LDIR
C615  017EFB    LD      BC,0FB7EH      ;registre d'état FDC
C618  3E03      LD      A,03H          ;code opération Specify Drive Param.
C61A  CD5CC9    CALL    0C95CH         ;envoyer au FDC
C61D  3A4ABE    LD      A,(0BE4AH)     ;délai d'attente en millisec. (12)
C620  3D        DEC     A
C621  07        RLCA
C622  07        RLCA
C623  07        RLCA
C624  2F        CPL
C625  E6F0      AND     0F0H           ;donne A0h = 12 ms Step Rate
C627  B6        OR      (HL)           ;Head Unload-Time dans les bits 0 à 3
C628  CD5CC9    CALL    0C95CH         ;envoyer au FDC
C62B  23        INC     HL
C62C  7E        LD      A,(HL)         ;Head Load Time
C62D  C35CC9    JP      0C95CH         ;Envoyer au FDC

*************** ^88 déterminer état lecteur
C630  CD38C6    CALL    0C638H         ;Routine déterminer état lecteur
C633  D0        RET     NC             ;=> erreur apparue
C634  3A4CBE    LD      A,(0BE4CH)     ;charger FDC-Status 0 dans accu
C637  C9        RET
```

```
*************** Routine déterminer état lecteur
C638  CD76C9    CALL    0C976H      ;Moteur marche, ajouter ticker pour
                                     délai
C63B  F5        PUSH    AF          ;Accu contient No lecteur
C63C  CD47C9    CALL    0C947H      ;Sense Interrupt Status FDC
C63F  017EFB    LD      BC,0FB7EH   ;registre d'état FDC
C642  3E04      LD      A,04H       ;code opération Sense Drive Status
C644  CD5CC9    CALL    0C95CH      ;envoyer accu au FDC
C647  F1        POP     AF          ;No lecteur
C648  CD5CC9    CALL    0C95CH      ;envoyer accu au FDC
C64B  C31CC9    JP      0C91CH      ;lire phase résultat FDC

*************** ^85  écrire secteur e=Drv, d=Trk, c=Sec, hl=buffer I/O
C64E  3E45      LD      A,45H       ;code d'opération écrire secteur
C650  1802      JR      0C654H

*************** ^86 formater piste
C652  3E4D      LD      A,4DH       ;code d'opération formater piste
C654  CD76C9    CALL    0C976H      ;Moteur marche, ajouter ticker pour
                                     délai
C657  0611      LD      B,11H       ;Nombre tentatives
C659  CD6DC6    CALL    0C66DH      ;Read/Write/Format cont'd
C65C  3A48BE    LD      A,(0BE48H)
C65F  3D        DEC     A
C660  03        INC     BC
C661  03        INC     BC
C662  03        INC     BC
C663  20FA      JR      NZ,0C65FH
C665  C9        RET

*************** ^84 lire secteur, e=Drv, d=Trk, c=Sec, hl=buffer I/O
C666  CD76C9    CALL    0C976H      ;Moteur marche, ajouter ticker pour
                                     délai
C669  3E66      LD      A,66H       ;code d'opération lire secteur
C66B  0610      LD      B,10H

*************** Read/Write/Format cont'd
C66D  2262BE    LD      (0BE62H),HL ;buffer I/O de 512 octets
C670  67        LD      H,A         ;code d'opération FDC
C671  69        LD      L,C         ;numéro secteur
C672  2274BE    LD      (0BE74H),HL ;ranger code opérat. et secteur voulu
C675  48        LD      C,B         ;Nombre des tentatives de lecture
```

```
C676  217CC6    LD      HL,0C67CH   ;programmer adresse FDC
C679  C3FFC6    JP      0C6FFH      ;chercher piste dans d, 'Call (hl)'

*************** programmer FDC pour action voulue
C67C  2A74BE    LD      HL,(0BE74H) ;(code opér. et secteur voulu)
C67F  017EFB    LD      BC,0FB7EH   ;registre d'état FDC
C682  7C        LD      A,H         ;code d'opération FDC
C683  CD5CC9    CALL    0C95CH      ;envoyer au FDC
C686  7B        LD      A,E         ;Head Sel./Unit Sel., donc No lecteur
C687  CD5CC9    CALL    0C95CH      ;envoyer au FDC
C68A  7C        LD      A,H         ;code d'opération sorti dans accu
C68B  FE4D      CP      4DH         ;formater piste?
C68D  2016      JR      NZ,0C6A5H   ;saut si lecture/écriture secteur
C68F  3E14      LD      A,14H
C691  CD59C9    CALL    0C959H      ;octets/secteur de bloc param.disc au
                                     FDC
C694  3E10      LD      A,10H
C696  CD59C9    CALL    0C959H      ;secteurs/piste de bloc param.disc au
                                     FDC
C699  3E12      LD      A,12H
C69B  CD59C9    CALL    0C959H      ;longueur GAP3 de bloc param. disc au
                                     FDC
C69E  3E13      LD      A,13H
C6A0  CD5CCA    CALL    0CA5CH      ;octet remplissage, valeur bloc
                                     param. lecteur 13h
C6A3  181C      JR      0C6C1H      ;Read/Write/Format Phase Execution

*************** entrée lecture/écriture d'un secteur
C6A5  7A        LD      A,D         ;numéro piste
C6A6  CD5CC9    CALL    0C95CH      ;envoyer accu au FDC
C6A9  AF        XOR     A           ;No tête (pour lecteur double-tête)
C6AA  CD5CC9    CALL    0C95CH      ;Envoyer accu au FDC
C6AD  7D        LD      A,L         ;Numéro secteur
C6AE  CD5CC9    CALL    0C95CH      ;Envoyer accu au FDC
C6B1  3E14      LD      A,14H       ;Octets/Secteur
C6B3  CD59C9    CALL    0C959H      ;tiré de bloc param. disque au FDC
C6B6  7D        LD      A,L         ;numéro secteur comme dernier secteur
C6B7  CD5CC9    CALL    0C95CH      ;Envoyer accu au FDC
C6BA  3E11      LD      A,11H       ;Longueur GAP3 pour Read/Write
C6BC  CD59C9    CALL    0C959H      ;tiré de bloc param. disque au FDC
C6BF  3EFF      LD      A,0FFH      ;DTL, doit être ffh
```

```
*************** Read/Write/Format Phase Execution
C6C1  CDD1C6    CALL    0C6D1H       ;Secteur I/O de 512 octets
C6C4  FB        EI
C6C5  CD07C9    CALL    0C907H       ;lire état FDC, Drive Ready Write
C6C8  D8        RET     C            ;Prot?
C6C9  C0        RET     NZ
C6CA  3A4DBE    LD      A,(0BE4DH)   ;Statusreg.1 FDC
C6CD  87        ADD     A,A
C6CE  D8        RET     C            ;=> Carry est OK
C6CF  AF        XOR     A
C6D0  C9        RET

*************** Secteur Read/Write, 512 Octets
C6D1  F3        DI
C6D2  CD5CC9    CALL    0C95CH       ;Envoyer accu au FDC
C6D5  7C        LD      A,H
C6D6  2A62BE    LD      HL,(0BE62H) ;buffer I/O de 512 octets
C6D9  FE66      CP      66H          ;Code d'opération lire secteur?
C6DB  2018      JR      NZ,0C6F5H    ;si non, à la boucle d'écriture
C6DD  1806      JR      0C6E5H       ;à la boucle de lecture

***** boucle de lecture, lire données jusque FDC annonce fin du secteur
C6DF  0C        INC     C            ;(bc) sur reg. données FDC
C6E0  ED78      IN      A,(C)        ;lire octet de données
C6E2  77        LD      (HL),A       ;sauver dans buffer (hl)
C6E3  0D        DEC     C            ;(bc) sur registre d'état FDC
C6E4  23        INC     HL           ;augmenter pointeur de buffer
C6E5  ED78      IN      A,(C)        ;retirer octet d'état
C6E7  F2E5C6    JP      P,0C6E5H     ;attendre octet message Ready
C6EA  E620      AND     20H          ;fin exécution, commencer résultat?
C6EC  20F1      JR      NZ,0C6DFH    ;retirer prochain octet
C6EE  C9        RET

*************** boucle d'écriture, écrire données jusque FDC annonce fin
C6EF  0C        INC     C            ;(bc) sur registre de données FDC
C6F0  7E        LD      A,(HL)       ;retirer octet du buffer
C6F1  ED79      OUT     (C),A        ;et écrire sur disque
C6F3  0D        DEC     C            ;(bc) sur registre d'état FDC
C6F4  23        INC     HL           ;augmenter pointeur de buffer
C6F5  ED78      IN      A,(C)        ;aller chercher octet d'état
C6F7  F2F5C6    JP      P,0C6F5H     ;prochain octet réclamé?
C6FA  E620      AND     20H          ;fin exécution, commencer résultat?
```

```
C6FC  20F1      JR     NZ,0C6EFH   ;écrire prochain octet
C6FE  C9        RET

*************** cherche la piste indiquée dans d
C6FF  3A66BE    LD     A,(0BE66H)  ;Nombre de tentatives de lecture
C702  47        LD     B,A
C703  CD2BC7    CALL   0C72BH      ;chercher piste
C706  D8        RET    C           ;=> trouvé piste
C707  2819      JR     Z,0C722H    ;=> 10 échecs, alors READ FAIL
C709  78        LD     A,B         ;nombre tentatives restant
C70A  E604      AND    04H
C70C  2809      JR     Z,0C717H    ;mettre flag recalibrate
C70E  D5        PUSH   DE          ;piste voulue
C70F  1627      LD     D,27H       ;piste 39
C711  CD66C7    CALL   0C766H      ;chercher piste
C714  D1        POP    DE          ;piste voulue
C715  18EC      JR     0C703H      ;nouvelle tentat. pour trouver piste

*************** fixer flag recalibrate sur recalibrate
C717  E5        PUSH   HL
C718  3E17      LD     A,17H       ;Octet 17h dans paramètre disc actuel
C71A  CD63CA    CALL   0CA63H      ;déterminer bloc, fixer flag
C71D  3600      LD     (HL),00H    ;recalibrate sur recalibrate
C71F  E1        POP    HL
C720  18E1      JR     0C703H

*************** sortir READ FAIL lors de recherche de piste
C722  79        LD     A,C
C723  C5        PUSH   BC
C724  CD7ACA    CALL   0CA7AH      ;lecteur dans c, sortir message, CIR
C727  C1        POP    BC
C728  20D5      JR     NZ,0C6FFH   ;chercher piste dans d, call (hl)
C72A  C9        RET

***************
C72B  CD54C7    CALL   0C754H      ;chercher piste dans d, CALL (HL)
C72E  D8        RET    C
C72F  C8        RET    Z
C730  CD47C9    CALL   0C947H      ;Sense Interrupt Status FDC
C733  CD54C7    CALL   0C754H      ;chercher piste dans d, CALL (HL)
C736  D8        RET    C
C737  C8        RET    Z
```

```
C738  7A        LD      A,D         ;numéro de piste
C739  FE27      CP      27H         ;40 pistes
C73B  05        DEC     B
C73C  300A      JR      NC,0C748H
C73E  04        INC     B
C73F  14        INC     D
C740  CD66C7    CALL    0C766H      ;chercher piste dans d
C743  15        DEC     D
C744  CD54C7    CALL    0C754H      ;chercher piste dans d, CALL (HL)
C747  D8        RET     C
C748  C8        RET     Z
C749  7A        LD      A,D         ;Numéro de piste
C74A  B7        OR      A           ;<>0?
C74B  2002      JR      NZ,0C74FH
C74D  05        DEC     B
C74E  C9        RET

***************
C74F  15        DEC     D
C750  CD66C7    CALL    0C766H      ;Chercher piste dans d
C753  14        INC     D
C754  CD66C7    CALL    0C766H      ;Chercher piste dans d
C757  E5        PUSH    HL
C758  C5        PUSH    BC
C759  CD1E00    CALL    001EH       ;CALL (HL)
C75C  C1        POP     BC
C75D  E1        POP     HL
C75E  D8        RET     C
C75F  20F3      JR      NZ,0C754H   ;Chercher piste dans d, CALL (HL)
C761  05        DEC     B
C762  C9        RET

*************** ^87 chercher piste dans registre d
C763  CD76C9    CALL    0C976H      ;Moteur en marche, ajouter ticker
                                     pour délai
C766  E5        PUSH    HL
C767  D5        PUSH    DE
C768  C5        PUSH    BC
C769  3A66BE    LD      A,(0BE66H)  ;Nombre de tentatives de lecture
C76C  47        LD      B,A
C76D  3E17      LD      A,17H       ;Octet 17h, flag recalibrate
C76F  CD63CA    CALL    0CA63H      ;du bloc param. disc actuel
```

```
C772  7E       LD    A,(HL)      ;dans accu
C773  B7       OR    A
C774  201F     JR    NZ,0C795H   ;=> pas de recalibrate
C776  C5       PUSH  BC          ;b:= Nombre tentatives de lecture
C777  017EFB   LD    BC,0FB7EH   ;registre d'état FDC
C77A  3E07     LD    A,07H       ;Recalibrate piste 0
C77C  CD5CC9   CALL  0C95CH      ;Envoyer accu au FDC
C77F  7B       LD    A,E         ;Head Select/Unit Select
C780  CD5CC9   CALL  0C95CH      ;Envoyer accu au FDC
C783  3E28     LD    A,28H       ;attendre 40 fois 12 millisecondes,
C785  CDC7C7   CALL  0C7C7H      ;puis lire FDC Interrupt Status
C788  302A     JR    NC,0C7B4H
C78A  3E16     LD    A,16H       ;Octet 16h, DPB actuel
C78C  CD63CA   CALL  0CA63H      ;c'est le numéro de piste actuel
C78F  3600     LD    (HL),00H    ;effacer
C791  23       INC   HL          ;Octet 17 dans bloc param. disc
C792  36FF     LD    (HL),0FFH   ;sur -1
C794  C1       POP   BC          ;b:= Nombre tentatives de lecture
C795  2B       DEC   HL
C796  7E       LD    A,(HL)      ;numéro de piste atteint
C797  92       SUB   D           ;comparer avec piste voulue
C798  2828     JR    Z,0C7C2H    ;=> position atteinte
C79A  C5       PUSH  BC          ;b:= Nombre tentatives de lecture
C79B  017EFB   LD    BC,0FB7EH   ;Registre d'états FDC
C79E  3E0F     LD    A,0FH       ;Code d'opération chercher piste
C7A0  CD5CC9   CALL  0C95CH      ;Envoyer accu au FDC
C7A3  7B       LD    A,E         ;Head Select/Unit Select
C7A4  CD5CC9   CALL  0C95CH      ;Envoyer accu au FDC
C7A7  7A       LD    A,D         ;Numéro de piste voulu
C7A8  CD5CC9   CALL  0C95CH      ;Envoyer accu au FDC
C7AB  96       SUB   (HL)        ;ôter numéro de piste atteinte
C7AC  3002     JR    NC,0C7B0H   ;Position atteinte
C7AE  7E       LD    A,(HL)
C7AF  92       SUB   D
C7B0  72       LD    (HL),D
C7B1  CDC7C7   CALL  0C7C7H      ;attendre, puis lire FDC Inter. Stat.
C7B4  C1       POP   BC
C7B5  380B     JR    C,0C7C2H    ;tout OK, piste trouvée
C7B7  20BD     JR    NZ,0C776H   ;nouvelle  tentative,  éventuellement
                                 avec Recalibrate
C7B9  05       DEC   B           ;compteur tentatives écoulé?
C7BA  CAADC9   JP    Z,0C9ADH    ;alors traitement erreur
```

```
C7BD  CD47C9    CALL   0C947H     ;Sense Interrupt Status FDC
C7C0  18B4      JR     0C776H     ;nouvelle tentative, éventuellement
                                   avec recalibrate

*************** piste a été trouvée
C7C2  C1        POP    BC
C7C3  D1        POP    DE
C7C4  E1        POP    HL
C7C5  37        SCF               ;marque que tout est OK
C7C6  C9        RET

*************** attend (Accu * 12)+ 16 ms, lit Int.Status FDC
C7C7  F5        PUSH   AF
C7C8  3A4ABE    LD     A,(0BE4AH)
C7CB  CDE0C7    CALL   0C7E0H     ;Boucle de tempor.
C7CE  F1        POP    AF
C7CF  3D        DEC    A
C7D0  20F5      JR     NZ,0C7C7H  ;Boucle de tempor. finie?
C7D2  3A49BE    LD     A,(0BE49H)
C7D5  CDE0C7    CALL   0C7E0H     ;Boucle de tempor.
C7D8  3E08      LD     A,08H      ;Code d'opér. Sense Interrupt Status
C7DA  CD5CC9    CALL   0C95CH     ;Envoyer accu au FDC
C7DD  C3F9C8    JP     0C8F9H     ;lire Int.Status FDC, Drive READY?

*************** attend accu*1 milliseconde
C7E0  F5        PUSH   AF         ;Accu est compteur de boucle
C7E1  3EF6      LD     A,0F6H     ;valeur de délai
C7E3  3D        DEC    A          ;compte à rebours accu jusqu'à 0
C7E4  20FD      JR     NZ,0C7E3H  ;environ 1 milliseconde
C7E6  F1        POP    AF         ;diminuer compteur de boucle
C7E7  3D        DEC    A
C7E8  20F6      JR     NZ,0C7E0H  ;éventuellement nouvelle boucle
C7EA  C9        RET

***************
C7EB  2153BE    LD     HL,0BE53H  ;buffer HS/US
C7EE  5E        LD     E,(HL)     ;numéro lecteur dans e
C7EF  3E03      LD     A,03H      ;valeur bloc param.disc 03h, masque
                                   bloc
C7F1  CD5CCA    CALL   0CA5CH     ;charger dans accu
C7F4  3C        INC    A          ;augmenter bloc masque
C7F5  1159BE    LD     DE,0BE59H
```

```
C7F8  12        LD     (DE),A        ;et ranger
C7F9  13        INC    DE            ;transférer lecteur, piste et numéro
C7FA  010300    LD     BC,0003H      ;enregistrement de be53h en be5ah
C7FD  EDB0      LDIR
C7FF  C9        RET

***************
C800  1159BE    LD     DE,0BE59H
C803  1A        LD     A,(DE)
C804  B7        OR     A
C805  C8        RET    Z
C806  13        INC    DE
C807  2153BE    LD     HL,0BE53H     ;buffer HS/US
C80A  0603      LD     B,03H         ;3 octets, lecteur, piste et secteur
C80C  1A        LD     A,(DE)        ;(hl) = (de)?
C80D  AE        XOR    (HL)
C80E  2006      JR     NZ,0C816H     ;=> différent, erreur
C810  13        INC    DE
C811  23        INC    HL
C812  10F8      DJNZ   0C80CH        ;prochain Octet
C814  37        SCF                  ;marque OK
C815  C9        RET

***************
C816  AF        XOR    A
C817  3259BE    LD     (0BE59H),A
C81A  C9        RET

************* encore place pour cet enregistrement? sinon piste suivante?
C81B  F5        PUSH   AF
C81C  2159BE    LD     HL,0BE59H
C81F  35        DEC    (HL)
C820  23        INC    HL
C821  5E        LD     E,(HL)        ;No lecteur dans e
C822  23        INC    HL
C823  23        INC    HL            ;No enregistrement dans la piste
C824  34        INC    (HL)          ;augmenter numéro enregistrement
C825  AF        XOR    A             ;octet faible enregistrements/piste
C826  CD5CCA    CALL   0CA5CH        ;de bloc param.disc dans accu
C829  BE        CP     (HL)          ;numéro enregistrement maxi atteint?
C82A  2004      JR     NZ,0C830H     ;=> pas encore atteint
C82C  3600      LD     (HL),00H      ;fixer enregistr. dans la piste sur 0
```

```
C82E  2B      DEC   HL          ;hl = be5b, numéro de piste
C82F  34      INC   (HL)        ;augmenter numéro de piste
C830  F1      POP   AF
C831  C9      RET

***************
C832  F5      PUSH  AF
C833  CD54C8  CALL  0C854H
C836  3819    JR    C,0C851H
C838  CD6FC8  CALL  0C86FH
C83B  C1      POP   BC          ;réparer pile pour cas d'erreur
C83C  D0      RET   NC          ;=> erreur
C83D  C5      PUSH  BC          ;pas erreur, bc à nouveau sur la pile
C83E  CD80C8  CALL  0C880H      ;teste si fin secteur
C841  F1      POP   AF
C842  3806    JR    C,0C84AH    ;=> enreg. déjà dans buffer secteur
C844  CDA2C8  CALL  0C8A2H      ;No secteur dans c,buffer sect.dns hl
C847  CD66C6  CALL  0C666H      ;lire secteur
C84A  F5      PUSH  AF          ;traiter erreur éventuelle
C84B  9F      SBC   A,A         ;ffh =pas d'erreur en lecture secteur
C84C  325EBE  LD    (0BE5EH),A
C84F  F1      POP   AF
C850  C9      RET

***************
C851  F1      POP   AF
C852  37      SCF
C853  C9      RET

***************
C854  3A5EBE  LD    A,(0BE5EH)  ;lire flag OK secteur
C857  B7      OR    A
C858  C8      RET   Z
C859  0153BE  LD    BC,0BE53H   ;buffer HS/US
C85C  2156BE  LD    HL,0BE56H
C85F  5E      LD    E,(HL)      ;No lecteur dans e
C860  0A      LD    A,(BC)      ;charger ancien No lecteur dans accu
C861  AE      XOR   (HL)        ;identiques?
C862  C0      RET   NZ          ;si non, alors RET
C863  03      INC   BC          ;ancien numéro de piste
C864  23      INC   HL          ;nouveau numéro de piste
C865  0A      LD    A,(BC)
```

```
C866  AE       XOR    (HL)         ;identiques?
C867  C0       RET    NZ           ;si non, alors retour
C868  CD92C8   CALL   0C892H       ;dépassement lors d'écriture enreg.
C86B  AE       XOR    (HL)
C86C  C0       RET    NZ           ;si dépassement, alors retour
C86D  37       SCF                 ;tout est OK
C86E  C9       RET

***************
C86F  215EBE   LD     HL,0BE5EH    ;annuler flag OK lire secteur
C872  3600     LD     (HL),00H
C874  2B       DEC    HL
C875  7E       LD     A,(HL)       ;(hl)=> flag écrire secteur
C876  B7       OR     A
C877  37       SCF
C878  C8       RET    Z            ;=> Flag à 0, ne pas écrire
C879  34       INC    (HL)
C87A  CDA2C8   CALL   0C8A2H       ;calculer No secteur effectif
C87D  C34EC6   JP     0C64EH       ;^85 écrire secteur

***************
C880  2156BE   LD     HL,0BE56H
C883  0153BE   LD     BC,0BE53H    ;buffer HS/US
C886  0A       LD     A,(BC)
C887  77       LD     (HL),A
C888  5F       LD     E,A          ;valeur Head Select/Unit Sel. dans e
C889  23       INC    HL
C88A  03       INC    BC
C88B  0A       LD     A,(BC)
C88C  77       LD     (HL),A
C88D  CD92C8   CALL   0C892H       ;détermine No sect.,teste si overflow
C890  77       LD     (HL),A
C891  C9       RET

*************** détermine No secteur, teste si dépassement enregistrement
C892  03       INC    BC           ;(numéro enregistrement)
C893  23       INC    HL
C894  3E15     LD     A,15H
C896  CD5CCA   CALL   0CA5CH       ;charger nombre enregistrements par
                                   secteur dans accu
```

```
C899  57         LD     D,A
C89A  0A         LD     A,(BC)       ;charger numéro enreg. dans accu
C89B  CB3A       SRL    D
C89D  D8         RET    C
C89E  CB3F       SRL    A
C8A0  18F9       JR     0C89BH

*************** calcule numéro secteur effectif
C8A2  ED5B56BE   LD     DE,(0BE56H) ;No lecteur et piste dans d et e
C8A6  3E0F       LD     A,0FH        ;charger dns accu bloc param.disc 0fh
C8A8  CD5CCA     CALL   0CA5CH       ;premier numéro secteur d'une piste
C8AB  2158BE     LD     HL,0BE58H    ;(numéro secteur voulu 0-8)
C8AE  86         ADD    A,(HL)       ;donne No du secteur à lire
C8AF  4F         LD     C,A          ;Numéro secteur dans c
C8B0  21B002     LD     HL,02B0H     ;déterminer adresse buffer secteur
C8B3  C39FCA     JP     0CA9FH       ;hl=hl+iy

*************** transférer enregistrement dans buffer secteur écriture
C8B6  E5         PUSH   HL           ;sauver tous les registres
C8B7  D5         PUSH   DE
C8B8  C5         PUSH   BC
C8B9  F5         PUSH   AF
C8BA  3EFF       LD     A,0FFH       ;écrire marque enregistrement
C8BC  325DBE     LD     (0BE5DH),A  ;Read/Write flag secteur
C8BF  CDD6C8     CALL   0C8D6H       ;alimenter bc, de et hl
C8C2  CD1BB9     CALL   0B91BH       ;KL LDIR,transférer enreg.dns secteur
C8C5  180A       JR     0C8D1H       ;restaurer tous les registres

** transférer enregistrement de buffer secteur dans buffer enregistrement
C8C7  E5         PUSH   HL           ;sauver tous les registres
C8C8  D5         PUSH   DE
C8C9  C5         PUSH   BC
C8CA  F5         PUSH   AF
C8CB  CDD6C8     CALL   0C8D6H       ;alimenter bc, de et hl
C8CE  EB         EX     DE,HL
C8CF  EDB0       LDIR                ;de buffer enreg. dans buffer secteur
C8D1  F1         POP    AF           ;restaurer tous les registres
C8D2  C1         POP    BC
C8D3  D1         POP    DE
C8D4  E1         POP    HL
C8D5  C9         RET
```

```
*************** de := début enreg. dans buffer secteur
C8D6  2153BE   LD     HL,0BE53H    ;buffer HS/US
C8D9  5E       LD     E,(HL)       ;Head Select/Unit Select dans e
C8DA  3E15     LD     A,15H        ;charger dns accu valeur bloc
                                    param.disc 15h
C8DC  CD5CCA   CALL   0CA5CH       ;nombre d'enregistrements par secteur
C8DF  3D       DEC    A
C8E0  23       INC    HL
C8E1  23       INC    HL
C8E2  A6       AND    (HL)         ;(numéro d'enregistrement)
C8E3  118000   LD     DE,0080H     ;longueur d'enregistrement 128 octets
C8E6  213002   LD     HL,0230H     ;décalage par rapport buffer enreg.
C8E9  3C       INC    A            ;correction nécessaire
C8EA  19       ADD    HL,DE        ;calcule adresse du prochain enreg.
C8EB  3D       DEC    A            ;dans buffer secteur
C8EC  20FC     JR     NZ,0C8EAH
C8EE  EB       EX     DE,HL        ;résultat dans hl
C8EF  CD98CA   CALL   0CA98H       ;de=de+iy, de => prochain enregistr.
C8F2  2A60BE   LD     HL,(0BE60H)  ;adresse buffer enreg. dans hl
C8F5  018000   LD     BC,0080H     ;taille enreg. 128 octets dans bc
C8F8  C9       RET

*************** lire registre d'état FDC, tester si DRIVE READY
C8F9  CD1CC9   CALL   0C91CH       ;lire phase résultat FDC
C8FC  D8       RET    C            ;=> aucune erreur apparue
C8FD  3A4CBE   LD     A,(0BE4CH)   ;registre d'état FDC 0
C900  E608     AND    08H          ;Drive Ready?
C902  C8       RET    Z            ;=> Drive est READY
C903  3E13     LD     A,13H        ;message d'erreur 13, Disc is missing
C905  180D     JR     0C914H

*********** lire registre d'état FDC, disquette protégée contre écriture
C907  CDF9C8   CALL   0C8F9H       ;phase résultat FDC, Drive READY?
C90A  D8       RET    C            ;=> tout est OK
C90B  C0       RET    NZ
C90C  3A4DBE   LD     A,(0BE4DH)   ;registre d'état FDC 1
C90F  E602     AND    02H          ;disquette protégée contr l'écriture?
C911  C8       RET    Z            ;=> n'est pas protégé
C912  3E12     LD     A,12H        ;mess.d'erreur12, Disc is write prot.
C914  CD7ACA   CALL   0CA7AH       ;Drive dans c, sortir message, CIR
C917  D8       RET    C            ;'Ignore'
C918  CAADC9   JP     Z,0C9ADH     ;'Cancel' => fin
```

```
C91B  C9        RET                 ;'Retry'

*************** lit et place octets de phase résultat FDC dans buffer
C91C  E5        PUSH    HL
C91D  D5        PUSH    DE
C91E  1600      LD      D,00H       ;nombre d'octets lus
C920  214CBE    LD      HL,0BE4CH   ;adresse buffer phase résultat
C923  E5        PUSH    HL
C924  ED78      IN      A,(C)       ;registre d'état FDC dans bc
C926  FEC0      CP      0C0H        ;attendre jusqu'à ce que
C928  38FA      JR      C,0C924H    ;octet d'état ready
C92A  0C        INC     C           ;adresse registre données FDC dans bc
C92B  ED78      IN      A,(C)       ;Interrupt-Statusregister
C92D  0D        DEC     C           ;adresse registre d'état FDC
C92E  77        LD      (HL),A      ;sauvegarder dans (hl)
C92F  23        INC     HL
C930  14        INC     D           ;compteur nombre octets retirés
C931  3E05      LD      A,05H       ;
C933  3D        DEC     A           ;courte boucle d'attente
C934  20FD      JR      NZ,0C933H   ;
C936  ED78      IN      A,(C)       ;lire octet d'état
C938  E610      AND     10H         ;Bit FDC Busy
C93A  20E8      JR      NZ,0C924H   ;instruction pas encore terminée
C93C  E1        POP     HL
C93D  7E        LD      A,(HL)
C93E  E6C0      AND     0C0H        ;isoler code interr., bits 6&7 de SR0
C940  2B        DEC     HL
C941  72        LD      (HL),D      ;sauver nombre octets phase résultat
C942  D1        POP     DE
C943  E1        POP     HL
C944  C0        RET     NZ          ;erreur apparue!
C945  37        SCF
C946  C9        RET                 ;terminé sans erreur

*************** Sense Interupt Status FDC
C947  C5        PUSH    BC
C948  017EFB    LD      BC,0FB7EH   ;registre d'état FDC
C94B  3E08      LD      A,08H       ;code d'opér. Sense Interrupt Status
C94D  CD5CC9    CALL    0C95CH      ;envoyer accu au FDC
C950  CD1CC9    CALL    0C91CH      ;lire phase résultat FDC
C953  FE80      CP      80H         ;message 'INVALID COMMAND'?
C955  20F4      JR      NZ,0C94BH   ;sinon essayer encore une fois
```

```
C957  C1        POP    BC
C958  C9        RET

************* charger dans accu octet du bloc param.disc (DPB) et sortir
C959  CD5CCA    CALL   0CA5CH      ;charger dans accu valeur DPB

*************** teste FDC, envoie accu à FDC s'il y a lieu
C95C  F5        PUSH   AF          ;valeur à sortir deux fois sur pile
C95D  F5        PUSH   AF
C95E  ED78      IN     A,(C)       ;Bit 7, tester Request for Master
C960  87        ADD    A,A
C961  30FB      JR     NC,0C95EH   ;attendre que nouvel octet réclamé
C963  87        ADD    A,A         ;Bit 6,direction données vers FDC
C964  3003      JR     NC,0C969H   ;Test si entrée ou sortie
C966  F1        POP    AF          ;ne pas envoyer octet au controller,
C967  F1        POP    AF          ;FDC enverra octet au processeur
C968  C9        RET

*************** accu est transmis au FDC
C969  F1        POP    AF          ;valeur à sortir de la pile
C96A  0C        INC    C           ;registre données FDC dans bc
C96B  ED79      OUT    (C),A       ;envoyer octet au FDC
C96D  0D        DEC    C           ;registre d'état FDC dans bc
C96E  3E05      LD     A,05H       ;
C970  3D        DEC    A           ;courte boucle d'attente
C971  00        NOP                ;compte à rebours accu de 5 à zéro
C972  20FC      JR     NZ,0C970H
C974  F1        POP    AF          ;restaurer valeur dans accu
C975  C9        RET

*************** moteur en marche, manipuler pile, (hl) sur buffer I/O
C976  2276BE    LD     (0BE76H),HL ;stockage provisoire
C979  E3        EX     (SP),HL     ;retirer adresse retour de la pile
C97A  D5        PUSH   DE          ;sauver bc et de
C97B  C5        PUSH   BC
C97C  ED7364BE  LD     (0BE64H),SP ;sauver pointeur de pile
C980  E5        PUSH   HL          ;adresse retour à nouveau sur la pile
C981  21ADC9    LD     HL,0C9ADH   ;
C984  E3        EX     (SP),HL     ;RETURN à nouveau -> hl,C9AD sur pile
C985  E5        PUSH   HL          ;adresse RETURN sur pile définitivem.
C986  D5        PUSH   DE          ;sauver de, bc, af,un RET après cette
C987  C5        PUSH   BC          ;routine conduit à C9AD!
```

```
C988  F5        PUSH    AF            ;pas simple, n'est-ce pas?
C989  CDDFC9    CALL    0C9DFH        ;Del Ticker
C98C  3A5FBE    LD      A,(0BE5FH)    ;flag moteur
C98F  B7        OR      A             ;tester
C990  2014      JR      NZ,0C9A6H     ;Moteur tourne déjà
C992  017EFA    LD      BC,0FA7EH     ;adresse port commande moteur
C995  3E01      LD      A,01H
C997  ED79      OUT     (C),A         ;mise en marche moteur
C999  ED5B44BE  LD      DE,(0BE44H)   ;nombre de ticks
C99D  CDCDC9    CALL    0C9CDH        ;appeler add ticker, délai plein rég.
C9A0  3A5FBE    LD      A,(0BE5FH)    ;flag moteur
C9A3  B7        OR      A             ;tester
C9A4  28FA      JR      Z,0C9A0H      ;attendre que flag moteur <> 0
C9A6  F1        POP     AF            ;retirer af, bc, de de pile
C9A7  C1        POP     BC
C9A8  D1        POP     DE
C9A9  2A76BE    LD      HL,(0BE76H)   ;(hl) pointe sur buffer I/O
C9AC  C9        RET                   ;le prochain RET va en C9AD!

***************

C9AD  ED7B64BE  LD      SP,(0BE64H)   ;rangé ici pour C97C
C9B1  F5        PUSH    AF
C9B2  ED5B46BE  LD      DE,(0BE46H)   ;nombre de ticks
C9B6  CDCDC9    CALL    0C9CDH        ;appeler Add Ticker
C9B9  F1        POP     AF            ;
C9BA  C1        POP     BC            ;bc, de et hl ont été PUSHés
C9BB  D1        POP     DE            ;vers C979 à C97B
C9BC  E1        POP     HL
C9BD  3E00      LD      A,00H
C9BF  D8        RET     C
C9C0  214CBE    LD      HL,0BE4CH
C9C3  7E        LD      A,(HL)
C9C4  E608      AND     08H
C9C6  23        INC     HL
C9C7  B6        OR      (HL)
C9C8  F640      OR      40H
C9CA  2B        DEC     HL
C9CB  2B        DEC     HL            ;(hl) => 0BE4B
C9CC  C9        RET

***************

C9CD  2167BE    LD      HL,0BE67H     ;Adresse Tick Block
```

```
C9D0  010000     LD      BC,0000H     ;Reload Count
C9D3  C3E9BC     JP      0BCE9H       ;KL ADD TICKER

*************** Routine Tick
C9D6  215FBE     LD      HL,0BE5FH    ;flag moteur
C9D9  7E         LD      A,(HL)       ;est 0 ou ffh
C9DA  2F         CPL                  ;devient ffh ou 0
C9DB  77         LD      (HL),A       ;sauvegarder
C9DC  B7         OR      A            ;si flag moteur zéro
C9DD  2806       JR      Z,0C9E5H     ;=> moteur tourne
C9DF  2167BE     LD      HL,0BE67H    ;sinon Adresse Tick Block
C9E2  C3ECBC     JP      0BCECH       ;KL DEL TICKER, déclencher ticker
                                      ;moteur
***************
C9E5  CDDFC9     CALL    0C9DFH
C9E8    3E00             LD                 A,00H              ;
C9EA  017EFA     LD      BC,0FA7EH    ;port moteur
C9ED  ED79       OUT     (C),A        ;restaur flipflop moteur,moteur coupé
C9EF  AF         XOR     A
C9F0  325FBE     LD      (0BE5FH),A   ;annuler flag moteur
C9F3  C9         RET

*************** organiser Disc Parameter Header et blocs DP
C9F4  212002     LD      HL,0220H     ;décalage pour DPH lecteur B
C9F7  11D001     LD      DE,01D0H     ;décalage pour DPB lecteur B
C9FA  CD03CA     CALL    0CA03H       ;initialiser DPH et DPB lecteur B
C9FD  211002     LD      HL,0210H     ;décalage pour DPH lecteur A
CA00  119001     LD      DE,0190H     ;décalage pour DPB lecteur A
CA03  CD98CA     CALL    0CA98H       ;de=de+iy, début du DPB
CA06  ED5342BE   LD      (0BE42H),DE  ;ranger adresse
CA0A  D5         PUSH    DE           ;et ranger sur la pile
CA0B  CD9FCA     CALL    0CA9FH       ;hl=hl+iy, début du DPH
CA0E  2240BE     LD      (0BE40H),HL  ;ranger également
CA11  E5         PUSH    HL           ;et sur la pile
CA12  2143CA     LD      HL,0CA43H    ;Adresse du DPB standard
CA15  011900     LD      BC,0019H     ;25 octets
CA18  EDB0       LDIR                 ;transfère à adresse correcte
CA1A  4B         LD      C,E          ;ranger début zone cheksum dans bc
CA1B  42         LD      B,D
CA1C  E1         POP     HL           ;début du DPH, soit XLT
CA1D  3600       LD      (HL),00H     ;conversion éventuelle facteur SKEW
CA1F  23         INC     HL           ;non utilisé, donc 0
```

```
CA20 3600           LD      (HL),00H
CA22 110700         LD      DE,0007H
CA25 19             ADD     HL,DE          ;(hl)=> DIRBUF
CA26 113002         LD      DE,0230H       ;décalage 128 octets pour buffer DIR
CA29 CD98CA         CALL    0CA98H         ;de=de+iy, adresse du buffer DIR
CA2C 73             LD      (HL),E         ;entrer dans DIRBUF
CA2D 23             INC     HL
CA2E 72             LD      (HL),D
CA2F 23             INC     HL
CA30 D1             POP     DE             ;début des DPBs
CA31 73             LD      (HL),E         ;entrer dans le header
CA32 23             INC     HL
CA33 72             LD      (HL),D
CA34 23             INC     HL             ;(hl)=> CSV, Checksum Vector
CA35 71             LD      (HL),C         ;(bc)=> adresse de zone checksum
CA36 23             INC     HL             ;entrer dans le header
CA37 70             LD      (HL),B
CA38 23             INC     HL             ;(hl)=> ALV, Allocation Vector
CA39 EB             EX      DE,HL          ;stockage provisoire dans de
CA3A 211000         LD      HL,0010H       ;décalage entre zone Checksum et
CA3D 09             ADD     HL,BC          ;zone allocation
CA3E EB             EX      DE,HL          ;(hl)=> ALV
CA3F 73             LD      (HL),E         ;entrer zone allocation dns le header
CA40 23             INC     HL
CA41 72             LD      (HL),D
CA42 C9             RET

*************** DPB standard bloc paramètres disque
CA43 2400           DEFW    24H            ;SPT enregistrements par piste
CA45 03             DEFB    03H            ;BSH Block Shift
CA46 07             DEFB    07H            ;BLM Block Mask
CA47 00             DEFB    00H            ;EXM Extent Mask
CA48 AA00           DEFW    00AAH          ;DSM nombre blocs libres -1
CA49 3F00           DEFW    003FH          ;DRM nombre entrées Dir -1
CA4B C000           DEFW    00C0H          ;ALO affectation Directory
CA4D 1000           DEFW    0010H          ;CKS nombre d'entrées à contrôler
CA4F 0200           DEFW    0002H          ;OFF décal. piste pour pistes système

*************** extensions des DPBs, pas prévu dans CP/M standard
CA4B 41             DEFB    41H            ;décalage secteur pour reconn. format
CA53 09             DEFB    09H            ;nombre secteurs/piste
CA54 2A             DEFB    2AH            ;longueur GAP3 pour Read/Write
```

```
CA55  52        DEFB    52H           ;longueur GAP3 pour formatage
CA56  E5        DEFB    E5H           ;octet remplissage pour formatage
CA57  02        DEFB    02H           ;octets/secteur pour FDC, 512 octets
CA58  04        DEFB    04H           ;nombre enregistrements par secteur
CA59  00        DEFB    00H           ;trois mémoires provisoires
CA5A  00        DEFB    00H
CA5B  00        DEFB    00H

*************** charger dans accu valeur DPB (A890h+(Drive*40h)+Accu)
CA5C  E5        PUSH    HL
CA5D  CD63CA    CALL    0CA63H
CA60  7E        LD      A,(HL)        ;Accu =(Table+(drive*40h)+Accu)
CA61  E1        POP     HL
CA62  C9        RET

*************** hl = Pointeur sur DPB actuel + accu
CA63  D5        PUSH    DE
CA64  2A42BE    LD      HL,(0BE42H) ;pointeur sur DPB lecteur 0
CA67  1D        DEC     E             ;Head Select/Unit Select -1
CA68  114000    LD      DE,0040H      ;décalage des DPBs lecteur 0 et 1
CA6B  2001      JR      NZ,0CA6EH     ;saut si drive 0 actuel
CA6D  19        ADD     HL,DE         ;déterminer début table lecteur 1
CA6E  5F        LD      E,A           ;Accu pointe sur octet voulu
CA6F  19        ADD     HL,DE         ;d=0, e=octet voulu
CA70  D1        POP     DE
CA71  C9        RET

*************** ^81h messages d'erreur Disc Controller on/off
CA72  2A78BE    LD      HL,(0BE78H) ;ranger ancienne valeur dans L
CA75  3278BE    LD      (0BE78H),A  ;entrer nouvelle valeur
CA78  7D        LD      A,L           ;transmettre ancienne valeur dns accu
CA79  C9        RET

*************** si autorisé, sortir message d'erreur
CA7A  F5        PUSH    AF            ;numéro d'erreur dans accu
CA7B  3A78BE    LD      A,(0BE78H)  ;Flag pour messages d'erreur
CA7E  B7        OR      A             ;tester
CA7F  2005      JR      NZ,0CA86H     ;=> pas de sortie autorisée
CA81  F1        POP     AF            ;répéter numéro d'erreur
CA82  4B        LD      C,E           ;numéro lecteur pour sortie dans C
CA83  C3B8CA    JP      0CAB8H        ;sortir message système, C, I or R
```

```
*************** fin pas de sortie autorisée
CA86  F1          POP     AF
CA87  AF          XOR     A            ;annuler accu et flags
CA88  C9          RET

*************** inutilisé jusqu'à ca90
CA89  FF          RST     38H
CA8A  FF          RST     38H
CA8B  FF          RST     38H
CA8C  FF          RST     38H
CA8D  FF          RST     38H
CA8E  FF          RST     38H
CA8F  FF          RST     38H

*************** BC = BC + IY
CA90  FDE5        PUSH    IY
CA92  E3          EX      (SP),HL      ;iy dans hl, hl sur pile
CA93  09          ADD     HL,BC        ;ajouter bc et hl
CA94  44          LD      B,H          ;résultat dans bc
CA95  4D          LD      C,L
CA96  E1          POP     HL           ;restaurer hl
CA97  C9          RET

*************** DE = DE + IY
CA98  FDE5        PUSH    IY
CA9A  E3          EX      (SP),HL      ;iy dans hl, hl sur pile
CA9B  19          ADD     HL,DE        ;ajouter de et hl
CA9C  EB          EX      DE,HL        ;résultat dans de
CA9D  E1          POP     HL           ;restaurer hl
CA9E  C9          RET

*************** HL = HL + IY
CA9F  D5          PUSH    DE
CAA0  FDE5        PUSH    IY
CAA2  D1          POP     DE           ;iy dans de
CAA3  19          ADD     HL,DE        ;ajouter hl et de
CAA4  D1          POP     DE           ;restaurer de
CAA5  C9          RET

*************** convertir minuscules en majuscules
CAA6  FE61        CP      61H          ;'a' ou supérieur?
CAA8  D8          RET     C
```

```
CAA9  FE7B      CP      7BH          ;'z' ou inférieur?
CAAB  D0        RET     NC
CAAC  C6E0      ADD     A,0E0H       ;convertir en majuscules
CAAE  C9        RET

*************** vider mémoire (de) à (de)+(bc)
CAAF  AF        XOR     A            ;annuler accu
CAB0  12        LD      (DE),A       ;vider mémoire
CAB1  13        INC     DE           ;prochaine adresse
CAB2  0B        DEC     BC           ;diminuer nombre
CAB3  78        LD      A,B          ;tester si bc = 0
CAB4  B1        OR      C
CAB5  20F8      JR      NZ,0CAAFH    ;vide (de) à (de+bc)
CAB7  C9        RET

*************** sortir erreur dans A, puis tester 'IGN, RET, CHAN'
CAB8  CDEBCA    CALL    0CAEBH       ;sortir message système dans a
CABB  3E14      LD      A,14H        ;message système 20
CABD  CDEBCA    CALL    0CAEBH       ;chercher et sortir
CAC0  CD09BB    CALL    0BB09H       ;KM READ CHAR
CAC3  38FB      JR      C,0CAC0H
CAC5  CD81BB    CALL    0BB81H       ;TXT CUR ON
CAC8  CD06BB    CALL    0BB06H       ;KM WAIT CHAR
CACB  CDA6CA    CALL    0CAA6H       ;convertir en majuscules
CACE  FE43      CP      43H          ;'C' = Cancel
CAD0  2811      JR      Z,0CAE3H     ;=> Z=1,C=0
CAD2  FE49      CP      49H          ;'I' = Ignore
CAD4  37        SCF                  ;=> Z=1,C=1
CAD5  280C      JR      Z,0CAE3H
CAD7  FE52      CP      52H          ;'R' =Retry
CAD9  2807      JR      Z,0CAE2H     ;=> Z=0,C=0
CADB  3E07      LD      A,07H        ;caractère 'BELL'
CADD  CD5ABB    CALL    0BB5AH       ;TXT OUTPUT, un bip
CAE0  18E6      JR      0CAC8H       ;attendre nouvelle entrée

*************** Entry Retry, annuler flags
CAE2  B7        OR      A

*************** Entry Cancel et Ignore, sortir caractère dans accu
CAE3  CD5ABB    CALL    0BB5AH       ;TXT OUTPUT
CAE6  CD84BB    CALL    0BB84H       ;TXT CUR OFF
CAE9  3E00      LD      A,00H        ;sortir 'CR/LF'
```

```
*************** MESSAGE SYSTEME, chercher message système et sortir
CAEB  E5        PUSH    HL
CAEC  C5        PUSH    BC
CAED  F5        PUSH    AF
CAEE  E67F      AND     7FH         ;annuler bit du numéro d'erreur
CAF0  2186CB    LD      HL,0CB86H   ;hl pointe sur mess.système et erreur
CAF3  47        LD      B,A         ;numéro d'erreur dans b
CAF4  04        INC     B           ;augmenter d'un, est immédiatement
CAF5  1805      JR      0CAFCH      ;diminué à nouveau lors de DJNZ

*************** ignorer messages jusqu'au message voulu
CAF7  7E        LD      A,(HL)      ;un caractère du message dans accu
CAF8  23        INC     HL          ;augmenter pointeur
CAF9  3C        INC     A           ;teste si fin d'un message
CAFA  20FB      JR      NZ,0CAF7H   ;si pas fin, continuer recherche
CAFC  10F9      DJNZ    0CAF7H      ;b<>0 ignorer prochain message
```

```
*************** sortir message voulu
CAFE  7E        LD     A,(HL)     ;un caractère du message dans accu
CAFF  23        INC    HL         ;augmenter pointeur
CB00  FEFF      CP     0FFH       ;fin du message atteinte?
CB02  280B      JR     Z,0CB0FH   ;si oui, alors saut
CB04  E5        PUSH   HL
CB05  D5        PUSH   DE
CB06  C5        PUSH   BC
CB07  CD13CB    CALL   0CB13H     ;continuer test car. dans a et sortir
CB0A  C1        POP    BC
CB0B  D1        POP    DE
CB0C  E1        POP    HL
CB0D  18EF      JR     0CAFEH     ;retirer prochain caractère

*************** message sorti, terminé
CB0F  F1        POP    AF
CB10  C1        POP    BC
CB11  E1        POP    HL
CB12  C9        RET

*************** continuer test caractère et sortir
CB13  B7        OR     A          ;tester caractère dans accu
CB14  F266CB    JP     P,0CB66H   ;inférieur 80h, alors à la sortie
CB17  FEFE      CP     0FEH       ;chaîne pour numéro de lecteur
CB19  2846      JR     Z,0CB61H   ;convert. No lecteur en A/B et sortir
CB1B  FEFC      CP     0FCH       ;chaîne pour variable numérique
CB1D  281A      JR     Z,0CB39H   ;déterminer variable et sortir
CB1F  FEFD      CP     0FDH       ;chaîne pour nom de fichier
CB21  20C8      JR     NZ,0CAEBH  ;sortir chaîne extens. messag système
CB23  0608      LD     B,08H      ;nom de fichier a 8 caractères
CB25  CD2FCB    CALL   0CB2FH     ;localiser dans mémoire et sortir
CB28  3E2E      LD     A,2EH      ;"."
CB2A  CD83CB    CALL   0CB83H     ;sortir
CB2D  0603      LD     B,03H      ;extension a trois octets de long
CB2F  13        INC    DE         ;de pointe sur position en mémoire
CB30  1A        LD     A,(DE)     ;caractère dans accu
CB31  E67F      AND    7FH        ;annuler bit 7
CB33  CD83CB    CALL   0CB83H     ;sortir
CB36  10F7      DJNZ   0CB2FH     ;nombre - 1, prochain caractère
CB38  C9        RET
```

```
***************
CB39  EB        EX     DE,HL        ;de pointe sur nom de fichier
CB3A  1620      LD     D,20H        ;" ", caractère espace
CB3C  019CFF    LD     BC,0FF9CH
CB3F  CD4DCB    CALL   0CB4DH
CB42  01F6FF    LD     BC,0FFF6H
CB45  CD4DCB    CALL   0CB4DH
CB48  7D        LD     A,L
CB49  C630      ADD    A,30H        ;décalage pour chiffre ASCII
CB4B  1836      JR     0CB83H       ;sortir

***************
CB4D  3EFF      LD     A,0FFH
CB4F  E5        PUSH   HL
CB50  3C        INC    A
CB51  09        ADD    HL,BC
CB52  3004      JR     NC,0CB58H
CB54  E3        EX     (SP),HL
CB55  E1        POP    HL
CB56  18F7      JR     0CB4FH

***************
CB58  E1        POP    HL
CB59  B7        OR     A            ;zéro dans accu?
CB5A  2802      JR     Z,0CB5EH     ;alors sortir espace
CB5C  1630      LD     D,30H        ;décalage pour chiffre ASCII
CB5E  82        ADD    A,D
CB5F  1822      JR     0CB83H       ;sortir

***************
CB61  79        LD     A,C          ;numéro de lecteur 0 ou 1
CB62  C641      ADD    A,41H        ;ajouter 'A'
CB64  181D      JR     0CB83H       ;sortir 'A' ou 'B'

*************** sortir caractère dans la fenêtre dans position curseur
CB66  F5        PUSH   AF           ;sauvegarde provisoire caractère
CB67  FE20      CP     20H          ;caractère espace?
CB69  2017      JR     NZ,0CB82H    ;alors sortir directement
CB6B  E5        PUSH   HL
CB6C  D5        PUSH   DE
CB6D  CD69BB    CALL   0BB69H       ;TXT GET WINDOW
CB70  CD78BB    CALL   0BB78H       ;TXT GET CURSOR
```

```
CB73  7A        LD    A,D         ;d contient colonne droite fenêtre
CB74  D604      SUB   04H
CB76  3F        CCF
CB77  3001      JR    NC,0CB7AH
CB79  BC        CP    H           ;h contient colonne curseur
CB7A  D1        POP   DE
CB7B  E1        POP   HL
CB7C  3004      JR    NC,0CB82H
CB7E  F1        POP   AF
CB7F  C3E9CA    JP    0CAE9H      ;sortir 'CR/LF'

***************
CB82  F1        POP   AF          ;caractère à nouveau dans accu
CB83  C35ABB    JP    0BB5AH      ;TXT OUTPUT

*************** messages d'erreur / messages système

*************** message système 0
CB86  0D 0A FF                    CR/LF

*************** message système 1
CB89  20 20 20 FF                 sortir trois espaces

*************** message système 2
CB8D  FC 4B FF                    'variable numérique'K

*************** message système 3
CB90  97 82 20 66 72 65 65 97     'CR/LF''CR/LF' 'variable num. K'
CB98  FF                          free

*************** message système 4
CB99  80 42 61 64 20 63 6F 6D     'CR/LF'Bad command'CR/LF'
CBA1  6D 61 6E 64 80 FF

*************** message système 5
CBA7  9B 61 6C 72 65 61 64 79     'CR/LF Filename' already exists
CBAF  20 65 78 69 73 74 73 80
CBB7  FF

*************** message système 6
CBB8  9B 6E 6F 74 20 66 6F 75     'CR/LF Filename' not found
CBC0  6E 64 80 FF
```

```
*************** message système 7
CBC4  95 64 69 72 65 63 74 6F       'CR/LF Drive A/B' directory 'full'
CBCC  72 79 20 9A FF

*************** message système 8
CBD1  98 9A FF                      'CR/LF Drive A/B Disc' 'full CR/LF'

*************** message système 9
CBD4  98 63 68 61 6E 67 65 64       'CR/LF Drive A/B Disc' changed,
CBDC  2C 20 63 6C 6F 73 69 6E       closing 'Filename' 'CR/LF'
CBE4  67 20 FD 80 FF

*************** message système A
CBE9  9B 69 73 20 9D 20 6F 6E       'CR/LF Filename' is 'read' only
CBF1  6C 79 80 FF

*************** message système B
CBF5  FD FF                         'Filename'

*************** message système C
CBF7  95 75 73 65 72 FC 80 FF       'CR/LF Drive A/B' user
                                    'variable numérique'
*************** message système D
CBFF  2E 2E 2E 5E 43 FF             ...^C

*************** message système E
CC05  96 43 50 2F 4D 80 FF          'CR/LF failed to load' CP/M 'CR/LF'

*************** message système F
CC0C  96 62 6F 6F 74 20 73 65       'CR/LF failed to load' boot sector
CC14  63 74 6F 72 80 FF             'CR/LF'

*************** message système 10
CC1A  95 9D 99 FF                   'CR/LF Drive A/B' 'read' ' fail
                                    CR/LF'
*************** message système 11
CC1E  95 9C 99 FF                   'CR/LF Drive A/B' 'write' ' fail
                                    CR/LF'
*************** message système 12
CC22  98 69 73 20 9C 20 70 72       'CR/LF Drive A/B disc is 'write'
CC2A  6F 74 65 63 74 65 64 80       protected 'CR/LF'
CC32  FF
```

```
*************** message système 13
CC33  98 6D 69 73 73 69 6E 67       'CR/LF Drive A/B disc 'missing
CC3B  80 FF'CR/LF'                  'CR/LF'

*************** message système 14
CC3D  80 52 65 74 72 79 2C 20       'CR/LF' Retry, Ignore or Cancel?
CC45  49 67 6E 6F 72 65 20 6F
CC4D  72 20 43 61 6E 63 65 6C
CC55  3F 20 FF

*************** message système 15
CC58  80 44 72 69 76 65 20 FE       'CR/LF' Drive 'A/B':
CC60  3A 20 FF

*************** message système 16
CC63  80 46 61 69 6C 65 64 20       'CR/LF' Failed to load
CC6B  74 6F 20 6C 6F 61 64 20
CC73  FF

*************** message système 17
CC74  80 80 FF                      'CR/LF' 'CR/LF'

*************** message système 18
CC77  95 64 69 73 63 20 FF          'CR/LF Drive A/B' disc

*************** message système 19
CC7E  20 66 61 6C 80 FF             fail 'CR/LF'

*************** message système 1A
CC85  66 75 6C 6C 80 FF             full 'CR/LF'

*************** message système 1B
CC8B  80 FD 20 FF                   'CR/LF' 'Filename'

*************** message système 1C
CC8F  77 72 69 74 65 FF             write

*************** message système 1D
CC95  72 65 61 64 FF                read

CC9A  FF          RST     38H
CC9B  FF          RST     38H
```

```
CC9C  FF        RST    38H
CC9D  FF        RST    38H
CC9E  FF        RST    38H
CC9F  FF        RST    38H

*************** détourner tous les vecteurs cassette pour le disque
CCA0  AF        XOR    A
CCA1  FD7700    LD     (IY+00H),A  ;Drive et User sur défaut A0
CCA4  FD7701    LD     (IY+01H),A
CCA7  3D        DEC    A           ;Accu = ffh
CCA8  FD7708    LD     (IY+08H),A  ;flag OPENIN actif sur inactif ffh
CCAB  FD772C    LD     (IY+2CH),A  ;flag OPENOUT actif sur inactif ffh<
CCAE  FD227DBE  LD     (0BE7DH),IY
CCB2  2177BC    LD     HL,0BC77H   ;vecteurs cassette
CCB5  116401    LD     DE,0164H    ;sauvegarder
CCB8  CD98CA    CALL   0CA98H      ;de=de+iy, mempool + 164h
CCBB  012700    LD     BC,0027H    ;13*3 octets = 13 vecteurs
CCBE  EDB0      LDIR               ;
CCC0  EB        EX     DE,HL
CCC1  3630      LD     (HL),30H    ;0CD30H est l'adresse d'entrée pour
CCC3  23        INC    HL          ;toutes les entrées CAS détournées
CCC4  36CD      LD     (HL),0CDH
CCC6  23        INC    HL
CCC7  CD12B9    CALL   0B912H      ;KL ASC CURR SELECTION, numéro de la
CCCA  77        LD     (HL),A      ;Rom disque comme 3ème octet pour RST
CCCB  3EC9      LD     A,0C9H      ;Code pour Return
CCCD  327FBE    LD     (0BE7FH),A  ;
CCD0  AF        XOR    A

*************** DISC
CCD1  CDE4CC    CALL   0CCE4H      ;Disc Out
CCD4  D0        RET    NC          ;erreur apparue alors RET

*************** DISC IN
CCD5  2177BC    LD     HL,0BC77H   ;Cass In Open
CCD8  0607      LD     B,07H       ;détourner Cass In Open et les
CCDA  CDE9CC    CALL   0CCE9H      ;six entrées suivantes
CCDD  D0        RET    NC
CCDE  219BBC    LD     HL,0BC9BH   ;Catalog
CCE1  04        INC    B
CCE2  1805      JR     0CCE9H      ;détourner Catalog
```

```
*************** DISC OUT
CCE4  218CBC    LD      HL,0BC8CH   ;Cass Out Open et les quatre entrées
CCE7  0605      LD      B,05H       ;suivantes

***************
CCE9  B7        OR      A           ;est-ce que des paramètres suivent?
CCEA  203F      JR      NZ,0CD2BH   ;si oui alors saut
CCEC  118B01    LD      DE,018BH    ;sinon détourner vecteurs voulus
CCEF  CD98CA    CALL    0CA98H      ;de=de+iy, discmem + 18bh
CCF2  36DF      LD      (HL),0DFH   ;Restart 3
CCF4  23        INC     HL
CCF5  73        LD      (HL),E      ;les entrées pointent toutes
CCF6  23        INC     HL          ;vers A88Bh
CCF7  72        LD      (HL),D
CCF8  23        INC     HL
CCF9  10F7      DJNZ    0CCF2H
CCFB  37        SCF
CCFC  C9        RET

*************** TAPE, restaurer vecteurs cassette
CCFD  CD18CD    CALL    0CD18H      ;restaurer Tape Out
CD00  D0        RET     NC          ;erreur apparue, alors RET

*************** TAPE IN
CD01  216401    LD      HL,0164H
CD04  1177BC    LD      DE,0BC77H
CD07  011500    LD      BC,0015H    ;7 vecteurs tous Tape In
CD0A  CD21CD    CALL    0CD21H
CD0D  D0        RET     NC          ;erreur apparue
CD0E  218801    LD      HL,0188H
CD11  119BBC    LD      DE,0BC9BH
CD14  0E03      LD      C,03H       ;un vecteur Cass Catalog
CD16  1809      JR      0CD21H

*************** TAPE OUT
CD18  217901    LD      HL,0179H
CD1B  118CBC    LD      DE,0BC8CH
CD1E  010F00    LD      BC,000FH    ;5 vecteurs tous Tape Out
CD21  B7        OR      A           ;est-ce que des paramètres suivent?
CD22  2007      JR      NZ,0CD2BH   ;alors sortir erreur
CD24  CD9FCA    CALL    0CA9FH      ;hl=hl+iy
CD27  EDB0      LDIR
```

```
CD29  37          SCF                        ;fixer Carry comme marque que OK
CD2A  C9          RET

***************
CD2B  3E04        LD      A,04H              ;message système 4, 'BAD#COMMAND'
CD2D  C3EBCA      JP      0CAEBH             ;chercher et sortir


*************** est appelé par toutes les entrées CAS au moyen de RST3
CD30  FD2A7DBE    LD      IY,(0BE7DH) ;début de mémoire pour disque dans iy
CD34  F3          DI                         ;nécessaire pour utilisation du
CD35  08          EX      AF,AF'             ;jeu de registres alternatif
CD36  D9          EXX
CD37  79          LD      A,C                ;contenu de c est variable
CD38  D1          POP     DE                 ;adresse de retour dans de
CD39  C1          POP     BC                 ;POPer deux autres RETs
CD3A  E1          POP     HL
CD3B  E3          EX      (SP),HL            ;ce RET doit remonter
CD3C  C5          PUSH    BC                 ;bc et RET original à nouvau sur pile
CD3D  D5          PUSH    DE
CD3E  4F          LD      C,A                ;restaurer c et b
CD3F  067F        LD      B,7FH
CD41  11D210      LD      DE,10D2H           ;augmenter adresse RET de 10D2h
CD44  19          ADD     HL,DE              ;et comme nouvelle adres.RET sur pile
CD45  E5          PUSH    HL                 ;RET pointe alors dans table suivante
CD46  D9          EXX                        ;répéter jeu de registre original
CD47  08          EX      AF,AF'
CD48  FB          EI                         ;les INTs sont à nouveau autorisées
CD49  C37FBE      JP      0BE7FH             ;il y a là un RET!

*************** bloc de jump pour les entrées CAS/DISC détournées
CD4C  C3AFCE      JP      0CEAFH             ;DISC IN OPEN
CD4F  C3B6D1      JP      0D1B6H             ;DISC IN CLOSE
CD52  C3BCD1      JP      0D1BCH             ;DISC IN ABANDON
CD55  C364CF      JP      0CF64H             ;DISC IN CHAR
CD58  C3F5CF      JP      0CFF5H             ;DISC IN DIRECT
CD5B  C369D0      JP      0D069H             ;DISC RETURN
CD5E  C365D0      JP      0D065H             ;DISC TEST EOF

CD61  C337CF      JP      0CF37H             ;DISC OUT OPEN
CD64  C3D8D1      JP      0D1D8H             ;DISC OUT CLOSE
CD67  C3C2D1      JP      0D1C2H             ;DISC OUT ABANDON
```

```
CD6A  C38FD0    JP      0D08FH      ;DISC OUT CHAR
CD6D  C3D8D0    JP      0D0D8H      ;DISC OUT DIRECT

CD70  C313D5    JP      0D513H      ;DISC CATALOG

*************** augmenter pointeur de pile et dans discmem+6
CD73  CD77CD    CALL    0CD77H      ;nécessaire pour correction de pile
CD76  C9        RET

*************** pointeur de pile dans discmem+6
CD77  E5        PUSH    HL
CD78  210600    LD      HL,0006H    ;deux CALLs et le PUSH HL = 6 octets
CD7B  39        ADD     HL,SP       ;pile corrigée comme il faut dans hl
CD7C  FD7506    LD      (IY+06H),L  ;et dans discmem+6
CD7F  FD7407    LD      (IY+07H),H  ;et ranger discmem+7
CD82  E1        POP     HL
CD83  C9        RET

************** sauvegarder pile, si OPENIN pas actif, alors interruption
CD84  CD77CD    CALL    0CD77H      ;savegarder pointeur de pile
CD87  F5        PUSH    AF
CD88  FD7E08    LD      A,(IY+08H)  ;flag OPENIN actif
CD8B  1807      JR      0CD94H

************** sauvegarder pile, si OPENOUT pas actif, alors interruption
CD8D  CD77CD    CALL    0CD77H      ;sauvegarder pointeur de pile
CD90  F5        PUSH    AF
CD91  FD7E2C    LD      A,(IY+2CH)  ;flag OPENOUT actif
CD94  FEFF      CP      0FFH        ;fichier pas ouvert?
CD96  2812      JR      Z,0CDAAH    ;=> interruption
CD98  CD16CE    CALL    0CE16H      ;si nécessaire Login, détermin format
CD9B  F1        POP     AF
CD9C  C9        RET

*************** interruption si OPENIN actif
CD9D  FD7E08    LD      A,(IY+08H)  ;flag OPENIN actif
CDA0  1803      JR      0CDA5H

*************** interruption si OPENOUT actif
CDA2  FD7E2C    LD      A,(IY+2CH)  ;flag OPENOUT actif
CDA5  CD77CD    CALL    0CD77H      ;sauvegarder pointeur de pile
CDA8  3C        INC     A           ;flag est ffh, si pas actif
```

```
CDA9  C8        RET    Z             ;=> n'était pas actif, tout est OK
CDAA  3E0E      LD     A,0EH         ;numéro d'erreur dans accu
CDAC  B7        OR     A             ;annuler flag Carry, marque erreur
CDAD  180A      JR     0CDB9H        ;restaurer pile, interrompre instruc.

*************** sortir Bad command, interrompre instruction
CDAF  3E04      LD     A,04H         ;Message système 4, 'Bad command'
CDB1  CDCADB    CALL   0DBCAH        ;
CDB4  C60C      ADD    A,0CH         ;numéro d'erreur dans accu
CDB6  F680      OR     80H           ;pas de message d'erreur jusqu'ici
CDB8  BF        CP     A             ;annuler Carry, marque erreur

*************** interrompre instruction
CDB9  FD6E06    LD     L,(IY+06H)    ;restaurer pointeur de pile
CDBC  FD6607    LD     H,(IY+07H)
CDBF  F9        LD     SP,HL
CDC0  C9        RET                  ;et retour

*************** teste si accu = 2, si non, interruption et 'Bad Command'
CDC1  3D        DEC    A

*************** teste si accu = 1, si non, interruption et 'Bad Command'
CDC2  3D        DEC    A
CDC3  C8        RET    Z             ;=> Accu est zéro
CDC4  C3AFCD    JP     0CDAFH        ;sortir Bad command, interrompre
                                     ;instruction
*************** amène longueur chaîne dans b, adresse chaîne dans hl
CDC7  CDCFCD    CALL   0CDCFH        ;amener un paramètre dans hl
CDCA  46        LD     B,(HL)        ;longueur de la chaîne
CDCB  23        INC    HL            ;(hl) => adresse de la chaîne
CDCC  C3F9DB    JP     0DBF9H        ;LD HL,(HL)

*************** amener un paramètre d'extension d'instruction dans hl
CDCF  DD6E00    LD     L,(IX+00H)    ;octets faible et
CDD2  DD6601    LD     H,(IX+01H)    ;fort du param. à transmettre dans hl
CDD5  DD23      INC    IX            ;ix sur paramètre suivant éventuel
CDD7  DD23      INC    IX
CDD9  C9        RET

*************** !A:
CDDA  AF        XOR    A             ;Accu sur 0, valeur pour lecteur A
CDDB  1802      JR     0CDDFH
```

```
*************** !B:
CDDD  3E01     LD     A,01H       ;Accu sur 1, valeur pour lecteur B
CDDF  CD73CD   CALL   0CD73H      ;sauvegarder pointeur de pile
CDE2  1813     JR     0CDF7H      ;transmettre valeur au DOS

*************** !DRIVE
CDE4  CD73CD   CALL   0CD73H      ;sauvegarder pile
CDE7  CDC2CD   CALL   0CDC2H      ;1 param.à suivre sinon 'Bad command'
CDEA  CDC7CD   CALL   0CDC7H      ;retirer paramètre
CDED  05       DEC    B
CDEE  C2AFCD   JP     NZ,0CDAFH   ;sortir Bad command,interrompre instr
CDF1  7E       LD     A,(HL)      ;marque lecteur voulue (A/B) dns accu
CDF2  CDA6CA   CALL   0CAA6H      ;convertir en majuscules
CDF5  D641     SUB    41H         ;donne 0 ou 1
CDF7  CD16CE   CALL   0CE16H      ;Login
CDFA  FD7700   LD     (IY+00H),A  ;transmettre No lecteur au DOS
CDFD  C9       RET

*************** !USER
CDFE  CD73CD   CALL   0CD73H      ;sauvegarder pile
CE01  CDC2CD   CALL   0CDC2H      ;1 param.à suivre sinon 'Bad command'
CE04  CDCFCD   CALL   0CDCFH      ;amener paramètre dans hl
CE07  111000   LD     DE,0010H    ;numéro User maximal + 1
CE0A  CDF3DB   CALL   0DBF3H      ;de = hl?, numéro User légal?
CE0D  D2AFCD   JP     NC,0CDAFH   ;trop grand, 'Bad command', interrup.
CE10  FD7501   LD     (IY+01H),L  ;transmettre numéro User au DOS
CE13  C9       RET

***************
CE14  0A       LD     A,(BC)      ;premier caractère extens. nom
CE15  03       INC    BC          ;de fichier, numéro lecteur
CE16  E5       PUSH   HL
CE17  D5       PUSH   DE
CE18  C5       PUSH   BC
CE19  F5       PUSH   AF
CE1A  4F       LD     C,A         ;No lecteur lecteur appelé
CE1B  1EFF     LD     E,0FFH      ;ffh = OPEN actif sur lecteur appelé
CE1D  FD7E08   LD     A,(IY+08H)  ;flag OPENIN actif
CE20  B9       CP     C           ;sur ce lecteur?
CE21  2808     JR     Z,0CE2BH    ;=> OPENIN actif
CE23  FD7E2C   LD     A,(IY+2CH)  ;flag OPENOUT actif
CE26  B9       CP     C           ;sur ce lecteur?
```

```
CE27  2802      JR     Z,0CE2BH     ;=> OPENOUT actif
CE29  1E00      LD     E,00H        ;00h = pas OPEN actif sur lecteur
CE2B  D5        PUSH   DE           ;appelé
CE2C  C5        PUSH   BC
CE2D  CDF0C4    CALL   0C4F0H       ;teste No lecteur,déterm. format disc
CE30  C1        POP    BC
CE31  D1        POP    DE
CE32  7C        LD     A,H          ;teste hl sur 0000
CE33  B5        OR     L            ;si hl = 0000,alors lecteur pas READY
CE34  CAAFCD    JP     Z,0CDAFH     ;sortir Bad command,interrompre instr
CE37  FD7503    LD     (IY+03H),L   ;(hl) => Disc Parameter Header
                                    ;(a910/a920)
CE3A  FD7404    LD     (IY+04H),H   ;dans iy+3/iy+4
CE3D  FD7305    LD     (IY+05H),E   ;flag, si OPEN actif sur lectr appelé
CE40  FD7102    LD     (IY+02H),C   ;numéro lecteur (HS/US)
CE43  F1        POP    AF
CE44  C1        POP    BC
CE45  D1        POP    DE
CE46  E1        POP    HL
CE47  C9        RET

*************** copie nom fichier étendu dans bloc header OPENIN
CE48  215000    LD     HL,0050H     ;décalage par rapp.bloc header OPENIN
CE4B  CD5ACE    CALL   0CE5AH
CE4E  E5        PUSH   HL
CE4F  114200    LD     DE,0042H
CE52  19        ADD    HL,DE
CE53  3680      LD     (HL),80H
```

```
CE55  E1          POP     HL
CE56  C9          RET

*************** copie nom de fichier étendu dans bloc header OPENOUT
CE57  219A00      LD      HL,009AH     ;décal. par rapp. bloc header OPENOUT
CE5A  C5          PUSH    BC
CE5B  D5          PUSH    DE
CE5C  CD9FCA      CALL    0CA9FH       ;hl=hl+iy, hl=> bloc header
CE5F  3600        LD      (HL),00H
CE61  23          INC     HL
CE62  73          LD      (HL),E       ;(de) => Adresse du buffer user
CE63  23          INC     HL
CE64  72          LD      (HL),D       ;dans pointeur début buffer user
CE65  23          INC     HL
CE66  73          LD      (HL),E       ;Vecteur dans buffer user
CE67  23          INC     HL           ;fixer sur début
CE68  72          LD      (HL),D
CE69  23          INC     HL           ;(hl)=> début nom fichier dans header
CE6A  E5          PUSH    HL
CE6B  C5          PUSH    BC           ;(bc) => EFN à partir numéro user
CE6C  014500      LD      BC,0045H     ;nombre d'octets à annuler
CE6F  EB          EX      DE,HL        ;
CE70  CDAFCA      CALL    0CAAFH       ;vide (de) à (de+bc), reste du header
CE73  C1          POP     BC           ;(bc) => EFN à partir numéro user
CE74  60          LD      H,B          ;transférer dans hl
CE75  69          LD      L,C
CE76  D1          POP     DE           ;adr. nom fichier dans header dans de
CE77  D5          PUSH    DE
CE78  010C00      LD      BC,000CH     ;longueur nom fichier avec Drive/User
CE7B  EDB0        LDIR                 ;tranférer dans bloc header
CE7D  E1          POP     HL           ;adr. nom fichier dans header dans hl
CE7E  D1          POP     DE           ;adresse buffer user dans de
CE7F  E5          PUSH    HL
CE80  011200      LD      BC,0012H
CE83  09          ADD     HL,BC
CE84  3616        LD      (HL),16H     ;marque type fichier 'unprot. ASCII'
CE86  23          INC     HL
CE87  23          INC     HL
CE88  23          INC     HL
CE89  73          LD      (HL),E       ;Adresse buffer user
CE8A  23          INC     HL
CE8B  72          LD      (HL),D
```

```
CE8C  23        INC    HL
CE8D  36FF      LD     (HL),0FFH
CE8F  E1        POP    HL
CE90  C1        POP    BC
CE91  C9        RET

*************** valeur contrôle deux octets sur header (43h octets)
CE92  E5        PUSH   HL          ;hl => début header
CE93  210000    LD     HL,0000H    ;fixer valeur contrôle sur 0
CE96  54        LD     D,H         ;octet fort de de mis à 0
CE97  0643      LD     B,43H       ;nombre d'octets
CE99  E3        EX     (SP),HL     ;hl = pointeur dans header, valeur
                                    contrôle sur pile
CE9A  7E        LD     A,(HL)      ;un caractère dans accu
CE9B  23        INC    HL          ;prochain caractère dans le header
CE9C  E3        EX     (SP),HL     ;hl = valeur de contrôle, pointeur
                                    dans header sur pile
CE9D  5F        LD     E,A         ;accu dans e, de = caract. en 16 bit
CE9E  19        ADD    HL,DE       ;ajouter à valeur de contrôle
CE9F  10F8      DJNZ   0CE99H      ;encore un caractère? =>
CEA1  EB        EX     DE,HL       ;valeur de contrôle dans de
CEA2  E1        POP    HL          ;hl => début header
CEA3  C9        RET

***************
CEA4  E5        PUSH   HL
CEA5  CD92CE    CALL   0CE92H
CEA8  73        LD     (HL),E
CEA9  23        INC    HL
CEAA  72        LD     (HL),D
CEAB  E1        POP    HL
CEAC  C3F9D3    JP     0D3F9H

*************** DISC IN OPEN
CEAF  CD9DCD    CALL   0CD9DH      ;ranger pointeur de pile
CEB2  D5        PUSH   DE          ;adresse buffer 2K
CEB3  CD6FDA    CALL   0DA6FH      ;contrôler validité nom de fichier
CEB6  CD14CE    CALL   0CE14H      ;header param. disc dans hl, Login
                                    s'il y a lieu
CEB9  210900    LD     HL,0009H
CEBC  09        ADD    HL,BC       ;(bc)=> nom de fichier étendu
CEBD  7E        LD     A,(HL)      ;(hl)=> premier caract. extension
```

```
CEBE  3C        INC     A             ;premier caract. extension =ffh?
CEBF  2808      JR      Z,0CEC9H      ;=> entré nom fichier sans extension
CEC1  CD51D6    CALL    0D651H        ;chercher nom de fichier dans Dir
CEC4  D20CD5    JP      NC,0D50CH     ;=> fichier pas trouvé, interruption
CEC7  181E      JR      0CEE7H

*************** entré nom de fichier sans extension
CEC9  CDA8D2    CALL    0D2A8H        ;entrer 3 espaces extension
CECC  CD51D6    CALL    0D651H        ;nom de fichier sur disque sans ext.?
CECF  3816      JR      C,0CEE7H      ;=> trouvé
CED1  CDB3D2    CALL    0D2B3H        ;entrer extension 'BAS'
CED4  CD51D6    CALL    0D651H        ;sauvegardé comme fichier Basic?
CED7  380E      JR      C,0CEE7H      ;=> trouvé
CED9  CDB7D2    CALL    0D2B7H        ;entrer extension 'BIN'
CEDC  CD51D6    CALL    0D651H        ;sauvegardé comme fichier binaire?
CEDF  F5        PUSH    AF
CEE0  D4A8D2    CALL    NC,0D2A8H     ;fichier pas trouvé, ext. 3 espaces
CEE3  F1        POP     AF
CEE4  D20CD5    JP      NC,0D50CH     ;=> interruption, 'File not found'

*************** trouvé nom de fichier dans le Directory
CEE7  D1        POP     DE            ;adresse buffer 2K
CEE8  CD48CE    CALL    0CE48H        ;copier nom fichier dns header OPENIN
CEEB  E5        PUSH    HL            ;adresse début header
CEEC  110800    LD      DE,0008H
CEEF  CD98CA    CALL    0CA98H        ;de=de+iy, adresse OPENIN FCB
CEF2  0B        DEC     BC
CEF3  0A        LD      A,(BC)        ;numéro de lecteur
CEF4  12        LD      (DE),A        ;dans OPENIN FCB
CEF5  CD9CD7    CALL    0D79CH        ;nb caract. dans fichier d'entrée à 0
CEF8  21E400    LD      HL,00E4H      ;
CEFB  CD9FCA    CALL    0CA9FH        ;hl=hl+iy, adresse buffer d'enregist.
CEFE  CD92D3    CALL    0D392H        ;enregistrement dans buffer d'enreg.
CF01  301F      JR      NC,0CF22H     ;=> erreur apparue
CF03  E5        PUSH    HL            ;(hl)=> début buffer d'enregistrement
CF04  D5        PUSH    DE            ;(de)=> nom de fichier dns OPENIN FCB
CF05  CD92CE    CALL    0CE92H        ;valeur contrôle 43h octets de
                                       l'enregistrement dans de
CF08  CDF9DB    CALL    0DBF9H        ;ld hl,(hl), valeur contr. sauvée
                                       éventuelle
CF0B  CDF3DB    CALL    0DBF3H        ;hl = de? si oui, enregistrement
                                       chargé est header
```

```
CF0E  D1          POP     DE
CF0F  E1          POP     HL
CF10  200D        JR      NZ,0CF1FH     ;=> valeur contrôle <>, fichier ASCII
CF12  115500      LD      DE,0055H
CF15  CD98CA      CALL    0CA98H        ;de=de+iy, bloc header OPENIN +5
CF18  014500      LD      BC,0045H      ;nombre d'octets header
CF1B  EDB0        LDIR                  ;transférer dans bloc header OPENIN
CF1D  1803        JR      0CF22H

*************** fichier d'entrée pas fichier ASCII
CF1F  CD9CD7      CALL    0D79CH        ;nb caract. dans fichier entrée sur 0
CF22  E1          POP     HL
CF23  E5          PUSH    HL
CF24  111500      LD      DE,0015H
CF27  19          ADD     HL,DE
CF28  5E          LD      E,(HL)        ;adresse, d'où fichier a été écrit
CF29  23          INC     HL            ;à l'origine, dans de
CF2A  56          LD      D,(HL)
CF2B  23          INC     HL
CF2C  23          INC     HL
CF2D  4E          LD      C,(HL)        ;longueur du fichier dans bc
CF2E  23          INC     HL
CF2F  46          LD      B,(HL)
CF30  E1          POP     HL
CF31  37          SCF                   ;marque OPENIN OK
CF32  9F          SBC     A,A           ;au système d'expl CPC pas d'erreur
CF33  FD7E67      LD      A,(IY+67H)    ;type de fichier de fichier ouvert
CF36  C9          RET

*************** DISC OUT OPEN
CF37  CDA2CD      CALL    0CDA2H        ;ranger pointeur de pile
CF3A  D5          PUSH    DE            ;adresse de buffer OPENOUT de 2K
CF3B  CD6ADA      CALL    0DA6AH        ;teste si nom fichier valide,
                                         organise EFN
CF3E  CD14CE      CALL    0CE14H        ;header param. disc dans hl, Login
                                         s'il y a lieu
CF41  D1          POP     DE            ;buffer OPENOUT 2K
CF42  CD57CE      CALL    0CE57H        ;copier EFN dans bloc header OPENOUT
CF45  E5          PUSH    HL            ;(hl) => header OPENOUT +5, No user
CF46  CDABD2      CALL    0D2ABH        ;entrer extension '$$$' dans 0HB
CF49  CD76D6      CALL    0D676H        ;fichier déjà sur la disquette?
CF4C  60          LD      H,B           ;adresse header OPENOUT dans hl
```

```
CF4D  69          LD      L,C
CF4E  2B          DEC     HL           ;(hl) => numéro lecteur
CF4F  112C00      LD      DE,002CH
CF52  CD98CA      CALL    0CA98H       ;de=de+iy, bloc fichier contr.OPENOUT
CF55  010D00      LD      BC,000DH     ;longueur nom fichier av. lect.& user
CF58  EDB0        LDIR                 ;transférer dans FCB OPENOUT
CF5A  011700      LD      BC,0017H     ;longueur affect.blocs dans Dir(10h)
                                        + pointeur (7)
CF5D  CDAFCA      CALL    0CAAFH       ;efface reste après nom de fichier
                                        dans FCB
CF60  E1          POP     HL           ;bloc header +5
CF61  37          SCF                  ;marque OPENOUT OK
CF62  9F          SBC     A,A          ;au système d'expl. CPC OPENOUT OK
CF63  C9          RET

*************** DISC IN CHAR
CF64  E5          PUSH    HL           ;sauver tous les registres
CF65  D5          PUSH    DE
CF66  C5          PUSH    BC
CF67  CD74CF      CALL    0CF74H       ;retirer caract. de buffer OPENIN
CF6A  C1          POP     BC           ;restaurer registres
CF6B  D1          POP     DE
CF6C  E1          POP     HL
CF6D  D0          RET     NC           ;marque erreur apparue
CF6E  FE1A        CP      1AH          ;lu marque EOF?
CF70  37          SCF                  ;marque que tout est OK
CF71  C0          RET     NZ           ;=> pas EOF
CF72  B7          OR      A            ;sinon annuler Carry
CF73  C9          RET                  ;et retour

*************** retire un caractère de fichier OPENIN ouvert
CF74  CD84CD      CALL    0CD84H       ;sauvegarder pile, tester flag OPENIN
CF77  FDE5        PUSH    IY           ;Disc-Mempool
CF79  D1          POP     DE           ;dans de
CF7A  215000      LD      HL,0050H
CF7D  19          ADD     HL,DE        ;(hl)=>marque In Char(1)/In Direct(2)
CF7E  7E          LD      A,(HL)       ;marque dans accu
CF7F  FE02        CP      02H          ;Disc In Direct était actif?
CF81  CAAACD      JP      Z,0CDAAH     ;alors erreur, interrupt.de l'instr.
CF84  3601        LD      (HL),01H     ;entrer marque Disc In Char
CF86  219500      LD      HL,0095H     ;tester, si caract. dans buffer
CF89  19          ADD     HL,DE
```

```
CF8A  7E      LD    A,(HL)     ;compteur de caractères de
CF8B  23      INC   HL         ;remplissage de trois octets
CF8C  B6      OR    (HL)
CF8D  23      INC   HL
CF8E  B6      OR    (HL)
CF8F  2836    JR    Z,0CFC7H   ;=> pas de caractère dans buffer
CF91  216800  LD    HL,0068H
CF94  19      ADD   HL,DE
CF95  7E      LD    A,(HL)
CF96  23      INC   HL
CF97  B6      OR    (HL)
CF98  2B      DEC   HL
CF99  CCCBCF  CALL  Z,0CFCBH   ;remplir buffer OPENIN de 2K
CF9C  7E      LD    A,(HL)     ;tester si nombre de caractères lus
CF9D  23      INC   HL         ;et placés dans buffer OPENIN = 0
CF9E  B6      OR    (HL)
CF9F  2826    JR    Z,0CFC7H   ;=> erreur, pas de caract.dans Buffer
CFA1  46      LD    B,(HL)     ;nombre octets lus dans bc
CFA2  2B      DEC   HL
CFA3  4E      LD    C,(HL)
CFA4  0B      DEC   BC         ;un caractère est lu, donc décompter
CFA5  71      LD    (HL),C     ;ranger nombre de caractères
CFA6  23      INC   HL         ;restant dans le buffer
CFA7  70      LD    (HL),B
CFA8  219500  LD    HL,0095H
CFAB  19      ADD   HL,DE
CFAC  0603    LD    B,03H
CFAE  7E      LD    A,(HL)
CFAF  D601    SUB   01H
CFB1  77      LD    (HL),A
CFB2  3003    JR    NC,0CFB7H
CFB4  23      INC   HL
CFB5  10F7    DJNZ  0CFAEH
CFB7  215300  LD    HL,0053H
CFBA  19      ADD   HL,DE
CFBB  5E      LD    E,(HL)     ;pointeur dans buffer OPENIN dans de
CFBC  23      INC   HL
CFBD  56      LD    D,(HL)
CFBE  EB      EX    DE,HL
CFBF  E7      RST   20H        ;LD A,(HL), retirer caract. de buffer
CFC0  EB      EX    DE,HL
CFC1  13      INC   DE         ;augmenter pointeur buffer OPENIN
```

```
CFC2  72        LD     (HL),D      ;et ranger
CFC3  2B        DEC    HL
CFC4  73        LD     (HL),E
CFC5  37        SCF                ;marque, un caractère retiré
CFC6  C9        RET

*************** conclusion erreur
CFC7  3E0F      LD     A,0FH       ;code erreur
CFC9  B7        OR     A           ;annuler Carry
CFCA  C9        RET

***************
CFCB  E5        PUSH   HL
CFCC  D5        PUSH   DE
CFCD  E5        PUSH   HL
CFCE  215100    LD     HL,0051H
CFD1  19        ADD    HL,DE
CFD2  CDF9DB    CALL   0DBF9H      ;ld hl,(hl), adresse buffer
CFD5  E5        PUSH   HL          ;ranger sur la pile
CFD6  011000    LD     BC,0010H    ;16 enregistrements
CFD9  CD49D0    CALL   0D049H      ;charger dns buffer OPENIN si présent
CFDC  3E10      LD     A,10H
CFDE  91        SUB    C           ;déterminer nombr enregistrements lus
CFDF  47        LD     B,A         ;nombre dans b
CFE0  0E00      LD     C,00H
CFE2  CB38      SRL    B           ;détermine nombre octets lus
CFE4  CB19      RR     C           ;résultat dans bc
CFE6  D1        POP    DE          ;pointeur buffer OPENIN
CFE7  E1        POP    HL          ;(hl)=> nombre octets lus
CFE8  71        LD     (HL),C      ;ranger
CFE9  23        INC    HL
CFEA  70        LD     (HL),B
CFEB  01EAFF    LD     BC,0FFEAH
CFEE  09        ADD    HL,BC
CFEF  73        LD     (HL),E      ;ranger pointeur buffer OPENIN
CFF0  23        INC    HL
CFF1  72        LD     (HL),D
CFF2  D1        POP    DE
CFF3  E1        POP    HL
CFF4  C9        RET
```

```
*************** DISC IN DIRECT
CFF5  CD84CD    CALL   0CD84H       ;sauver pile, tester flag OPENIN
CFF8  E5        PUSH   HL           ;adresse de chargement
CFF9  215000    LD     HL,0050H     ;
CFFC  CD9FCA    CALL   0CA9FH       ;hl=hl+iy
CFFF  7E        LD     A,(HL)       ;tester marque In Char(1)/In Direct
D000  FE01      CP     01H          ;si Disc in Char,
D002  CAAACD    JP     Z,0CDAAH     ;erreur,interruption de l'instruction
D005  3602      LD     (HL),02H     ;entrer marque Disc In Direct
D007  114500    LD     DE,0045H
D00A  19        ADD    HL,DE
D00B  5E        LD     E,(HL)       ;nombre Caractères dans de
D00C  23        INC    HL
D00D  56        LD     D,(HL)
D00E  E1        POP    HL
D00F  D5        PUSH   DE
D010  E5        PUSH   HL           ;adresse de chargement dans hl
D011  EB        EX     DE,HL        ;et échanger
D012  3E07      LD     A,07H        ;2^7=128
D014  CDEBDB    CALL   0DBEBH       ;divise nombre caractères/128
D017  44        LD     B,H          ;résultat nombre enreg. dans bc
D018  4D        LD     C,L
D019  E1        POP    HL           ;adresse de chargement dans hl
D01A  CD49D0    CALL   0D049H       ;charger nombre enregistr. calculé
D01D  D1        POP    DE
D01E  301E      JR     NC,0D03EH    ;=> erreur
D020  7B        LD     A,E
D021  E67F      AND    7FH
D023  2819      JR     Z,0D03EH
D025  F5        PUSH   AF
D026  E5        PUSH   HL
D027  21E400    LD     HL,00E4H
D02A  CD9FCA    CALL   0CA9FH       ;hl=hl+iy
D02D  E5        PUSH   HL
D02E  010100    LD     BC,0001H
D031  CD49D0    CALL   0D049H
D034  E1        POP    HL
D035  D1        POP    DE
D036  C1        POP    BC
D037  3005      JR     NC,0D03EH
D039  48        LD     C,B
D03A  0600      LD     B,00H
```

```
D03C  EDB0      LDIR
D03E  216F00    LD      HL,006FH
D041  CD9FCA    CALL    0CA9FH
D044  37        SCF
D045  9F        SBC     A,A
D046  C3F9DB    JP      0DBF9H        ;ld hl,(hl)

***************
D049  1814      JR      0D05FH

*************** enregistrements dans buffer OPENIN, nombre enreg. dans bc
D04B  CD92D3    CALL    0D392H        ;enreg., évent. de disc, dans buffer
D04E  D0        RET     NC            ;=> fin fichier au autre erreur
D04F  116700    LD      DE,0067H
D052  CD98CA    CALL    0CA98H        ;de=de+iy, adresse type de fichier
                                       dans header OPENIN
D055  1A        LD      A,(DE)        ;tester type de fichier
D056  1F        RRA
D057  DC52D2    CALL    C,0D252H      ;=> Carry que pour 'Protected File'
D05A  118000    LD      DE,0080H      ;longueur d'enregistrement
D05D  19        ADD     HL,DE         ;augmenter pointeur de buffer
D05E  0B        DEC     BC            ;nombre d'enregistrem. encore à lire
D05F  78        LD      A,B           ;tester si nombre = 0
D060  B1        OR      C
D061  20E8      JR      NZ,0D04BH     ;=> pas encore tous lus
D063  37        SCF                   ;tout est OK, lu tous les enreg.
D064  C9        RET

*************** DISC TEST EOF
D065  CD64CF    CALL    0CF64H        ;appeler Disc In Char
D068  D0        RET     NC            ;RET si EOF, sinon renvoyer caractère
                                      ;dans buffer
*************** DISC RETURN
D069  E5        PUSH    HL
D06A  D5        PUSH    DE
D06B  F5        PUSH    AF
D06C  215300    LD      HL,0053H
D06F  CD9FCA    CALL    0CA9FH        ;hl=hl+iy
D072  5E        LD      E,(HL)
D073  23        INC     HL
D074  56        LD      D,(HL)
D075  1B        DEC     DE
```

```
D076  72        LD     (HL),D
D077  2B        DEC    HL
D078  73        LD     (HL),E
D079  54        LD     D,H
D07A  5D        LD     E,L
D07B  214200    LD     HL,0042H
D07E  19        ADD    HL,DE
D07F  CDABD7    CALL   0D7ABH
D082  211500    LD     HL,0015H
D085  19        ADD    HL,DE
D086  34        INC    (HL)
D087  2002      JR     NZ,0D08BH
D089  23        INC    HL
D08A  34        INC    (HL)
D08B  F1        POP    AF
D08C  D1        POP    DE
D08D  E1        POP    HL
D08E  C9        RET

*************** DISC OUT CHAR
D08F  CD8DCD    CALL   0CD8DH     ;range pile, teste flag OPENOUT actif
D092  E5        PUSH   HL
D093  D5        PUSH   DE
D094  C5        PUSH   BC
D095  F5        PUSH   AF         ;accu contient le caractère
D096  FDE5      PUSH   IY         ;Disc-Mempool dans de
D098  D1        POP    DE
D099  219A00    LD     HL,009AH
D09C  19        ADD    HL,DE      ;flag Disc Out Mode (Char/Direct)
D09D  7E        LD     A,(HL)     ;de header dans accu
D09E  FE02      CP     02H        ;jusqu'ici Disc Out Direct?
D0A0  CAAACD    JP     Z,0CDAAH   ;=> fin de l'instruction
D0A3  3601      LD     (HL),01H   ;entrer marque Disc Out Char
D0A5  21B200    LD     HL,00B2H
D0A8  19        ADD    HL,DE      ;(hl):= nombre car.dns buffer OPENOUT
D0A9  E5        PUSH   HL
D0AA  CDF9DB    CALL   0DBF9H     ;ld hl,(hl), nbre caractères dans hl
D0AD  0100F8    LD     BC,0F800H
D0B0  09        ADD    HL,BC
D0B1  D5        PUSH   DE
D0B2  DC18D1    CALL   C,0D118H   ;=> plus de2K,écrire données sur disc
D0B5  D1        POP    DE
```

```
D0B6  E1        POP    HL
D0B7  34        INC    (HL)          ;augmenter nmbre car.dans buffer user
D0B8  23        INC    HL            ;valeur deux octets
D0B9  2001      JR     NZ,0D0BCH     ;n'augmenter octet fort
D0BB  34        INC    (HL)          ;que si nécessaire
D0BC  21DF00    LD     HL,00DFH
D0BF  19        ADD    HL,DE
D0C0  CDABD7    CALL   0D7ABH        ;augmenter File Character Counter
D0C3  219D00    LD     HL,009DH
D0C6  19        ADD    HL,DE         ;(hl) => vecteur sur pointeur dans
D0C7  F1        POP    AF            ;buffer user
D0C8  4E        LD     C,(HL)        ;bc => pointeur dans buffer user
D0C9  23        INC    HL
D0CA  46        LD     B,(HL)
D0CB  2B        DEC    HL
D0CC  02        LD     (BC),A        ;placer caractère dans accu dans
                                      buffer user
D0CD  34        INC    (HL)          ;augmenter pointeur dans buffer user
D0CE  2002      JR     NZ,0D0D2H     ;également octet fort si nécessaire
D0D0  23        INC    HL
D0D1  34        INC    (HL)
D0D2  C1        POP    BC
D0D3  D1        POP    DE
D0D4  E1        POP    HL
D0D5  37        SCF                  ;marque que tout est OK
D0D6  9F        SBC    A,A           ;message au système CPC: OK
D0D7  C9        RET

*************** DISC OUT DIRECT
D0D8  CD8DCD    CALL   0CD8DH        ;range pile, teste flag OPENOUT actif
D0DB  F5        PUSH   AF            ;accu = type de fichier
D0DC  E5        PUSH   HL            ;adresse, à partir de laquelle écrire
D0DD  D5        PUSH   DE            ;longueur du bloc de données
D0DE  219A00    LD     HL,009AH
D0E1  CD9FCA    CALL   0CA9FH        ;hl=hl+iy, Disc Out Mode(Char/Direct)
D0E4  7E        LD     A,(HL)        ;dans accu
D0E5  FE01      CP     01H           ;Disc Out Char actif jusqu'ici?
D0E7  CAAACD    JP     Z,0CDAAH      ;alors erreur, fin de l'instruction
D0EA  3602      LD     (HL),02H      ;entrer marque direct
D0EC  112000    LD     DE,0020H
```

```
D0EF  19          ADD     HL,DE
D0F0  70          LD      (HL),B       ;bc = adresse entrée
D0F1  2B          DEC     HL
D0F2  71          LD      (HL),C
D0F3  C1          POP     BC           ;bc = longueur bloc de données
D0F4  2B          DEC     HL
D0F5  70          LD      (HL),B
D0F6  2B          DEC     HL
D0F7  71          LD      (HL),C
D0F8  112900      LD      DE,0029H
D0FB  19          ADD     HL,DE
D0FC  70          LD      (HL),B
D0FD  2B          DEC     HL
D0FE  71          LD      (HL),C
D0FF  11D3FF      LD      DE,0FFD3H
D102  19          ADD     HL,DE
D103  71          LD      (HL),C
D104  23          INC     HL
D105  70          LD      (HL),B
D106  C1          POP     BC
D107  23          INC     HL
D108  71          LD      (HL),C
D109  23          INC     HL
D10A  70          LD      (HL),B
D10B  11E6FF      LD      DE,0FFE6H
D10E  19          ADD     HL,DE
D10F  71          LD      (HL),C
D110  23          INC     HL
D111  70          LD      (HL),B
D112  F1          POP     AF           ;Type de fichier dans accu
D113  111500      LD      DE,0015H
D116  19          ADD     HL,DE
D117  77          LD      (HL),A       ;Type fichier dns bloc header OPENOUT

*************** écrire 2K (buffer user avec OUT CHAR) sur la disquette
D118  FDE5        PUSH    IY           ;Disc Mempool
D11A  D1          POP     DE           ;dans de
D11B  21B600      LD      HL,00B6H
D11E  19          ADD     HL,DE
D11F  7E          LD      A,(HL)       ;octet header 'First Block'
D120  B7          OR      A
D121  2818        JR      Z,0D13BH     ;=> pas le premier bloc
```

```
D123  21B100   LD     HL,00B1H
D126  19       ADD    HL,DE       ;(hl):= Type de fichier
D127  7E       LD     A,(HL)      ;Type de fichier dans accu,
D128  E60F     AND    0FH         ;Hi-Nibble est indifférent
D12A  FE06     CP     06H         ;marque 'unprotected ASCII'?
D12C  280D     JR     Z,0D13BH    ;=> c'est le cas
D12E  212C00   LD     HL,002CH
D131  19       ADD    HL,DE       ,hl=> bloc contrôle fichier OPENOUT
D132  D5       PUSH   DE          ;Disc-Mempool
D133  EB       EX     DE,HL
D134  CDA7D7   CALL   0D7A7H      ;Nombre enreg. +1 pour enreg. header
D137  CD7DD7   CALL   0D77DH      ;tester nombre enreg. dans FCB
D13A  D1       POP    DE
D13B  21B200   LD     HL,00B2H
D13E  19       ADD    HL,DE
D13F  E5       PUSH   HL
D140  5E       LD     E,(HL)      ;Block Char Counter,
D141  23       INC    HL          ;Nombre des caractères dans de
D142  56       LD     D,(HL)
D143  01E8FF   LD     BC,0FFE8H
D146  09       ADD    HL,BC       ;hl => User Buffer Vector
D147  CDF9DB   CALL   0DBF9H      ;ld hl,(hl), hl => User Buffer
D14A  E5       PUSH   HL
D14B  CD64D1   CALL   0D164H      ;Buffer dans enreg., alors sur Disque
D14E  C1       POP    BC          ;bc => User Buffer
D14F  E1       POP    HL          ;hl => Character Count
D150  3600     LD     (HL),00H    ;mettre à 0
D152  23       INC    HL
D153  3600     LD     (HL),00H
D155  23       INC    HL
D156  23       INC    HL
D157  23       INC    HL
D158  3600     LD     (HL),00H
D15A  11E7FF   LD     DE,0FFE7H
D15D  19       ADD    HL,DE
D15E  71       LD     (HL),C
D15F  23       INC    HL
D160  70       LD     (HL),B
D161  37       SCF                ;Marque OK
D162  9F       SBC    A,A
D163  C9       RET
```

```
***************
D164  D5       PUSH  DE        ;Nombre caractères dans bloc
D165  3E07     LD    A,07H     ;Nombre enreg./bloc
D167  EB       EX    DE,HL     ;Nombre dans de
D168  CDEBDB   CALL  0DBEBH    ;divise nombre car.s /128
D16B  EB       EX    DE,HL     ;résultat nécessitait enreg. dans de
D16C  42       LD    B,D       ;et bc
D16D  4B       LD    C,E
D16E  CD88D1   CALL  0D188H    ;transférer enregistrements
D171  C1       POP   BC        ;Nombre caractères
D172  79       LD    A,C       ;octet faible dans accu
D173  E67F     AND   7FH       ;limiter à un enregistrement
D175  C8       RET   Z         ;si zéro, alors pas d'autres octets
D176  4F       LD    C,A       ;sinon reste fichier dans nouvel enr.
D177  0600     LD    B,00H     ;alors octet fort = zéro
D179  11E400   LD    DE,00E4H
D17C  CD98CA   CALL  0CA98H    ;de=de+iy
D17F  D5       PUSH  DE        ;buffer enregistrement
D180  CD1BB9   CALL  0B91BH    ;KL LDIR,transf.dernier enr.dns buffe
D183  3E1A     LD    A,1AH     ;marque EOF
D185  12       LD    (DE),A    ;ajouter
D186  E1       POP   HL        ;hl => buffer enregistrement
D187  03       INC   BC        ;bc = augmenter compteur d'enregistr.
D188  1827     JR    0D1B1H

*************** écrire enreg. (Nombre dans bc) dans fichier
D18A  E5       PUSH  HL
D18B  11B100   LD    DE,00B1H
D18E  CD98CA   CALL  0CA98H    ;de=de+iy
D191  1A       LD    A,(DE)    ;Type de fichier dans accu
D192  1F       RRA             ;Bit 0 mis?
D193  3013     JR    NC,0D1A8H ;=> pas mis, pas 'Protected'
D195  C5       PUSH  BC
D196  11E400   LD    DE,00E4H
D199  CD98CA   CALL  0CA98H    ;de=de+iy
D19C  D5       PUSH  DE        ;de = buffer enregistrement
D19D  018000   LD    BC,0080H  ;longueur d'enregistrement
D1A0  CD1BB9   CALL  0B91BH    ;KL LDIR, bloc header dns buffer enr.
D1A3  E1       POP   HL
D1A4  C1       POP   BC
D1A5  CD52D2   CALL  0D252H    ;'protéger' enregistrement
D1A8  CDAFD3   CALL  0D3AFH    ;enreg.dns buffer secteur, évtl disc
```

```
D1AB  E1        POP   HL              ;hl => buffer enregistrement
D1AC  118000    LD    DE,0080H        ;longueur d'enregistrement
D1AF  19        ADD   HL,DE
D1B0  0B        DEC   BC              ;diminuer nombre des enregistrements
D1B1  78        LD    A,B             ;tous enreg. écrits?
D1B2  B1        OR    C
D1B3  20D5      JR    NZ,0D18AH       ;=> encore enreg. à écrire
D1B5  C9        RET

*************** DISC dans CLOSE
D1B6  CD84CD    CALL  0CD84H          ;sauvegarder pile, tester flag OPENIN
D1B9  CDE5C9    CALL  0C9E5H          ;Moteur éteint, déclencher Event

*************** DISC dans ABANDON
D1BC  FD3608FF  LD    (IY+08H),0FF;mettre flag OPENIN actif sur inactif
D1C0  186E      JR    0D230H          ;fin

*************** DISC OUT ABANDON
D1C2  CD8DCD    CALL  0CD8DH          ;sauvegarde pile, teste flag OPENOUT
D1C5  112D00    LD    DE,002DH
D1C8  CD98CA    CALL  0CA98H          ;de=de+iy
D1CB  AF        XOR   A
D1CC  CD3CD8    CALL  0D83CH          ;libérer à nouveau blocs dans
D1CF  1B        DEC   DE              ;table d'affectation
D1D0  3EFF      LD    A,0FFH
D1D2  12        LD    (DE),A
D1D3  CD1FC5    CALL  0C51FH          ;chercher piste 0
D1D6  1858      JR    0D230H          ;fin

*************** DISC OUT CLOSE
D1D8  21DF00    LD    HL,00DFH        ;File Character Count
D1DB  CD9FCA    CALL  0CA9FH          ;hl=hl+iy
D1DE  7E        LD    A,(HL)          ;teste si des caractères ont été
D1DF  23        INC   HL              ;transférés,
D1E0  B6        OR    (HL)
D1E1  23        INC   HL
D1E2  B6        OR    (HL)
D1E3  28DD      JR    Z,0D1C2H        ;sinonDiscOutAbandon,pas d'entrée DIR
D1E5  CD8DCD    CALL  0CD8DH          ;sauver pointeur pile, tester flag
                                       OPENOUT
D1E8  CD18D1    CALL  0D118H          ;transf. dernier enreg. dans buffer
D1EB  112C00    LD    DE,002CH
```

```
D1EE  CD98CA   CALL   0CA98H      ;de=de+iy, nom fichier dans
D1F1  D5       PUSH   DE          ;FCB OPENOUT
D1F2  CD8CD7   CALL   0D78CH      ;entrer nom fichier et affectation
D1F5  019F00   LD     BC,009FH    ;blocs dans dir
D1F8  CD90CA   CALL   0CA90H      ;bc=bc+iy, bc:= bloc header OPENOUT
D1FB  211200   LD     HL,0012H
D1FE  09       ADD    HL,BC       ;(hl):= Type de fichier
D1FF  5E       LD     E,(HL)
D200  210900   LD     HL,0009H
D203  09       ADD    HL,BC
D204  7E       LD     A,(HL)      ;1er caractère extension dans accu
D205  3C       INC    A           ;tester si ffh
D206  2016     JR     NZ,0D21EH   ;=> extension indiquée
D208  7B       LD     A,E         ;Type de fichier dans accu
D209  E60E     AND    0EH
D20B  2005     JR     NZ,0D212H
D20D  CDB3D2   CALL   0D2B3H      ;entrer extension 'BAS'
D210  180C     JR     0D21EH

***************
D212  FE02     CP     02H         ;Type de fichier 2?
D214  2005     JR     NZ,0D21BH
D216  CDB7D2   CALL   0D2B7H      ;entrer extension 'BIN'
D219  1803     JR     0D21EH

***************
D21B  CDA8D2   CALL   0D2A8H      ;entrer extension 3 espaces
D21E  60       LD     H,B         ;Adresse bloc header OPENOUT dans hl
D21F  69       LD     L,C
D220  7B       LD     A,E         ;Type de fichier dans accu
D221  E60F     AND    0FH         ;Hi-Nibble est indifférent
D223  FE06     CP     06H         ;'Unprotected ASCII'?
D225  C4A4CE   CALL   NZ,0CEA4H   ;=> est 'protected'
D228  C1       POP    BC          ;FCB OPENOUT dans bc
D229  3EFF     LD     A,0FFH
D22B  02       LD     (BC),A      ;Marque, OPENOUT pas actif
D22C  03       INC    BC
D22D  CDDAD2   CALL   0D2DAH      ;remplace '$$$' par extension orig.
D230  37       SCF                ;Marque DISC OUT CLOSE OK
D231  9F       SBC    A,A         ;message au système CPC: tout est OK
D232  C9       RET
```

```
***************
D233  FD6602    LD      H,(IY+02H)
D236  FD360500  LD      (IY+05H),00H ;pas OPEN actif sur lecteur appelé
D23A  110800    LD      DE,0008H
D23D  CD43D2    CALL    0D243H
D240  112C00    LD      DE,002CH
D243  CD98CA    CALL    0CA98H       ;de=de+iy
D246  1A        LD      A,(DE)
D247  BC        CP      H
D248  C0        RET     NZ
D249  3EFF      LD      A,0FFH
D24B  12        LD      (DE),A
D24C  13        INC     DE
D24D  3E09      LD      A,09H
D24F  C3CADB    JP      0DBCAH

***************
D252  E5        PUSH    HL
D253  C5        PUSH    BC
D254  E5        PUSH    HL
D255  110101    LD      DE,0101H
D258  0681      LD      B,81H
D25A  180E      JR      0D26AH

*************** ;'Protected File', protection par XOR
D25C  E3        EX      (SP),HL      ;pointeur buffer OPENIN dans hl
D25D  E7        RST     20H          ;RAM LAM, caract. de buffer dans accu
D25E  E3        EX      (SP),HL
D25F  AE        XOR     (HL)
D260  DDAE00    XOR     (IX+00H)
D263  E3        EX      (SP),HL
D264  77        LD      (HL),A       ;retour octet dans buffer OPENIN
D265  23        INC     HL           ;augmenter pointeur buffer
D266  E3        EX      (SP),HL      ;et à nouveau sur la pile
D267  DD23      INC     IX           ;prochain octet XOR
D269  23        INC     HL           ;prochain octet XOR
D26A  15        DEC     D            ;compteur octets pour table ix
D26B  2006      JR      NZ,0D273H    ;=> table pas encore terminée
D26D  160B      LD      D,0BH        ;table 11 octets pour ix
D26F  DD2181D2  LD      IX,0D281H    ;début table dans ix
D273  1D        DEC     E            ;compteur octets pour table hl
D274  2005      JR      NZ,0D27BH    ;=> table pas encore terminée
```

```
D276  1E0D     LD     E,0DH      ;table 13 octets pour hI
D278  218CD2   LD     HL,0D28CH  ;début table
D27B  10DF     DJNZ   0D25CH     ;b:= Nombre des octets à coder
D27D  E1       POP    HL
D27E  D1       POP    DE
D27F  E1       POP    HL
D280  C9       RET

*************** table pour ix
D281  49 B1 36 F0 2E 1E
D287  06 2A 28 19 EA

*************** table pour hl
D28C  E2 9D DB 1A 42 29
D292  39 C6 B3 C6 90 45
D298  8A

*************** extensions
D299  202020   DEFM   '   '
D29C  242424   DEFM   '$$$'
D29F  42414B   DEFM   'BAK'
D2A2  424153   DEFM   'BAS'
D2A5  42494E   DEFM   'BIN'

***************
D2A8  AF       XOR    A          ;trois espaces
D2A9  180E     JR     0D2B9H

***************
D2AB  3E03     LD     A,03H      ;'$$$' pour nom fichier temporaire
D2AD  180A     JR     0D2B9H

***************
D2AF  3E06     LD     A,06H      ;'BAK' pour fichier backup
D2B1  1806     JR     0D2B9H

***************
D2B3  3E09     LD     A,09H      ;'BAS' pour fichiers Basic
D2B5  1802     JR     0D2B9H

***************
D2B7  3E0C     LD     A,0CH      ;'BIN' pour fichiers binaires
```

```
D2B9  D5        PUSH   DE
D2BA  C699      ADD    A,99H        ;ajouter octet faible de début table
D2BC  5F        LD     E,A          ;résultat dans e
D2BD  CED2      ADC    A,0D2H       ;ajouter octet fort avec Carry
D2BF  93        SUB    E            ;soustraire octet faible
D2C0  57        LD     D,A          ;octet fort résultat dans d
D2C1  1807      JR     0D2CAH

*************** entrer extension dans nom fichier
D2C3  D5        PUSH   DE           ;
D2C4  11A800    LD     DE,00A8H
D2C7  CD98CA    CALL   0CA98H       ;de=de+iy
D2CA  E5        PUSH   HL
D2CB  C5        PUSH   BC
D2CC  210900    LD     HL,0009H     ;longueur nom fichier avec User,
D2CF  09        ADD    HL,BC        ;sans extension
D2D0  010300    LD     BC,0003H     ;longueur extension
D2D3  EB        EX     DE,HL
D2D4  EDB0      LDIR                ;entrer extens.voulue dns nom fichier
D2D6  C1        POP    BC
D2D7  E1        POP    HL
D2D8  D1        POP    DE
D2D9  C9        RET

***************
D2DA  210C00    LD     HL,000CH     ;longueur nom fichier + extension
D2DD  09        ADD    HL,BC
D2DE  36FF      LD     (HL),0FFH
D2E0  23        INC    HL
D2E1  23        INC    HL
D2E2  36FF      LD     (HL),0FFH
D2E4  CD83D6    CALL   0D683H       ;nbre entrées Dir et tab affect sur 0
D2E7  E5        PUSH   HL
D2E8  210000    LD     HL,0000H
D2EB  E3        EX     (SP),HL
D2EC  CDA2D6    CALL   0D6A2H       ;prochaine entrée DIR dans (de)
D2EF  E3        EX     (SP),HL
D2F0  3028      JR     NC,0D31AH    ;=> pas d'autre entrée dans DIR
D2F2  CDAFD2    CALL   0D2AFH       ;ajouter extension 'BAK'
D2F5  CDD8D7    CALL   0D7D8H       ;cherche nom fichier indiqué sur disc
D2F8  3008      JR     NC,0D302H    ;=> BAK pas présent
D2FA  2601      LD     H,01H
```

```
D2FC  CDD9D9    CALL   0D9D9H      ;teste,si fichier READ ONLY
D2FF  3801      JR     C,0D302H    ;=> est READ ONLY
D301  24        INC    H
D302  CDC3D2    CALL   0D2C3H      ;entrer extension dans (de)
D305  CDD8D7    CALL   0D7D8H      ;cherche nom indiqué sur disque
D308  3008      JR     NC,0D312H   ;=> fichier pas présent
D30A  2E01      LD     L,01H
D30C  CDD9D9    CALL   0D9D9H      ;teste, si fichier READ ONLY
D30F  3801      JR     C,0D312H    ;=> fichier est READ ONLY
D311  2C        INC    L
D312  7C        LD     A,H
D313  B7        OR     A
D314  28D5      JR     Z,0D2EBH
D316  7D        LD     A,L
D317  B7        OR     A
D318  28D1      JR     Z,0D2EBH
D31A  F1        POP    AF
D31B  7D        LD     A,L
D31C  B7        OR     A
D31D  2843      JR     Z,0D362H
D31F  3D        DEC    A
D320  2866      JR     Z,0D388H
D322  7C        LD     A,H
D323  B7        OR     A
D324  283C      JR     Z,0D362H
D326  3D        DEC    A
D327  2845      JR     Z,0D36EH
D329  CD83D6    CALL   0D683H      ;nbre entrées Dir & tab affect sur 0
D32C  CDA2D6    CALL   0D6A2H
D32F  D0        RET    NC
D330  CD35D3    CALL   0D335H
D333  18F7      JR     0D32CH

***************
D335  CDAFD2    CALL   0D2AFH      ;entrer extension 'BAK'
D338  CDD8D7    CALL   0D7D8H      ;cherche nom fichier indiqué sur disc
D33B  DAAAD4    JP     C,0D4AAH    ;=> trouvé
D33E  CD51D3    CALL   0D351H      ;chercher fichier '$$$' dans Dir
D341  D8        RET    C           ;=> trouvé
D342  CDC3D2    CALL   0D2C3H      ;ajouter extension d'origine
D345  CDD8D7    CALL   0D7D8H      ;cherche nom fichier indiqué sur disc
D348  D0        RET    NC          ;=> pas trouvé
```

```
D349  C5        PUSH   BC
D34A  42        LD     B,D
D34B  4B        LD     C,E
D34C  CDAFD2    CALL   0D2AFH      ;entrer extension 'BAK'
D34F  180D      JR     0D35EH

*************** teste, si nom fichier avec '$$$' est déjà sur disquette
D351  CDABD2    CALL   0D2ABH      ;entrer extension '$$$'
D354  CDD8D7    CALL   0D7D8H      ;cherche nom fichier indiqué dans Dir
D357  D0        RET    NC          ;=> pas trouvé
D358  C5        PUSH   BC          ;(bc) => nom fichier
D359  42        LD     B,D         ;adresse de l'entrée Dir identique
D35A  4B        LD     C,E         ;dans bc
D35B  CDC3D2    CALL   0D2C3H      ;écrire extension d'origine dans
D35E  C1        POP    BC          ;enregistrement Dir
D35F  C37AD9    JP     0D97AH

***************
D362  CD83D6    CALL   0D683H      ;Nbre entrées Dir & tab affect sur 0
D365  CDA2D6    CALL   0D6A2H      ;déterminer proch.entrée Dir, de=nbre
D368  D0        RET    NC          ;=> pas d'autre entrée Dir
D369  CD3ED3    CALL   0D33EH      ;cherche noms fich. temp($$$)et orig.
D36C  18F7      JR     0D365H      ;continuer recherche

***************
D36E  CD83D6    CALL   0D683H      ;Nbre entrées Dir & tab affect sur 0
D371  CDA2D6    CALL   0D6A2H      ;déterminer proch.entrée Dir, de=nbre
D374  D0        RET    NC          ;=> pas d'autre entrée Dir
D375  CD7AD3    CALL   0D37AH      ;si fichier temp. trouvé, ext.orig.
                                    dans enregistrement
D378  18F7      JR     0D371H      ;sinon continuer recherche

***************
D37A  CD51D3    CALL   0D351H
D37D  D8        RET    C
D37E  CDC3D2    CALL   0D2C3H      ;entrer extension dans (de)
D381  CDD8D7    CALL   0D7D8H      ;cherche nom fichier indiqué sur disc
D384  DAAAD4    JP     C,0D4AAH
D387  C9        RET

*************** fichier READ ONLY, interrompre extension
D388  CDC3D2    CALL   0D2C3H      ;entrer extension dans (de)
```

```
D38B  50        LD     D,B
D38C  59        LD     E,C
D38D  3E0A      LD     A,0AH       ;mess.systèe 10, fichier read only
D38F  C3B1CD    JP     0CDB1H      ;sortir, terminer instruction

***************
D392  E5        PUSH   HL
D393  D5        PUSH   DE
D394  C5        PUSH   BC
D395  E5        PUSH   HL
D396  110800    LD     DE,0008H
D399  CD98CA    CALL   0CA98H      ;de=de+iy, adresse FCB OPENIN
D39C  CD10D4    CALL   0D410H      ;teste, si dernier enregistrement lu
D39F  3008      JR     NC,0D3A9H   ;=> pas d'autre enregistr.dns fichier
D3A1  EB        EX     DE,HL       ;de:=No enreg., hl:= nom fich dns FCB
D3A2  E3        EX     (SP),HL     ;(hl):= buffer enregistrement
D3A3  CDE8D9    CALL   0D9E8H      ;retirer enregistr.dans buffer enreg.
D3A6  D1        POP    DE
D3A7  1848      JR     0D3F1H

***************
D3A9  E1        POP    HL
D3AA  C1        POP    BC
D3AB  D1        POP    DE
D3AC  E1        POP    HL
D3AD  B7        OR     A
D3AE  C9        RET

*************** enreg. dans buffer secteur, éventuellement sur disque
D3AF  E5        PUSH   HL          ;buffer enregistrement
D3B0  D5        PUSH   DE
D3B1  C5        PUSH   BC          ;Nombre enreg.
D3B2  E5        PUSH   HL          ;et encore buffer enregistrement
D3B3  112C00    LD     DE,002CH
D3B6  CD98CA    CALL   0CA98H      ;de=de+iy, FCB OPENOUT
D3B9  CDC8D6    CALL   0D6C8H      ;teste si DISK FULL
D3BC  380B      JR     C,0D3C9H    ;=> encore place
D3BE  3E08      LD     A,08H       ;message système 8, disc full
D3C0  C2B1CD    JP     NZ,0CDB1H   ;sortir, interrompre instruction
D3C3  CD8CD7    CALL   0D78CH      ;cherche entrée Dir libre
D3C6  CDFAD6    CALL   0D6FAH
D3C9  CD2FD7    CALL   0D72FH
```

```
D3CC  0E00     LD     C,00H
D3CE  3818     JR     C,0D3E8H
D3D0  D5       PUSH   DE
D3D1  EB       EX     DE,HL
D3D2  CD93D8   CALL   0D893H      ;chercher bloc libre et occuper
D3D5  EB       EX     DE,HL
D3D6  3E08     LD     A,08H       ;sortir message système 8, disc full
D3D8  D2B1CD   JP     NC,0CDB1H   ;interrompre instr. si pas bloc libre
```

```
D3DB  73      LD     (HL),E      ;ranger bloc occupé
D3DC  78      LD     A,B
D3DD  B7      OR     A
D3DE  2802    JR     Z,0D3E2H
D3E0  23      INC    HL
D3E1  72      LD     (HL),D
D3E2  D1      POP    DE
D3E3  CD2FD7  CALL   0D72FH
D3E6  0E02    LD     C,02H
D3E8  EB      EX     DE,HL
D3E9  E3      EX     (SP),HL
D3EA  CDF3D9  CALL   0D9F3H      ;enregistrement dans buffer secteur
D3ED  D1      POP    DE          ;éventuellement sur disque
D3EE  CD7DD7  CALL   0D77DH
D3F1  CDA7D7  CALL   0D7A7H      ;nombre enregistrements +1
D3F4  C1      POP    BC
D3F5  D1      POP    DE
D3F6  E1      POP    HL
D3F7  37      SCF                ;marque que tout est OK
D3F8  C9      RET

***************
D3F9  E5      PUSH   HL
D3FA  112C00  LD     DE,002CH
D3FD  CD98CA  CALL   0CA98H
D400  CD9CD7  CALL   0D79CH
D403  CD10D4  CALL   0D410H
D406  EB      EX     DE,HL
D407  E1      POP    HL
D408  0E00    LD     C,00H
D40A  DAF3D9  JP     C,0D9F3H    ;enregistrement dans buffer secteur,
                                  éventuellement sur disque
D40D  C3AFCD  JP     0CDAFH      ;sortirBad command, interrompre
                                 ;instruction
*************** teste, si dernier enregistrement a été lu
D410  CDC8D6  CALL   0D6C8H
D413  3812    JR     C,0D427H
D415  C0      RET    NZ
D416  CDFAD6  CALL   0D6FAH
D419  D5      PUSH   DE
D41A  42      LD     B,D
D41B  4B      LD     C,E
```

```
D41C  03          INC    BC
D41D  C5          PUSH   BC
D41E  CDB3D7      CALL   0D7B3H
D421  EB          EX     DE,HL
D422  D1          POP    DE
D423  DCDFDB      CALL   C,0DBDFH     ;32 octets de (hl) dans (de)
D426  D1          POP    DE
D427  DC0CD7      CALL   C,0D70CH
D42A  DA2FD7      JP     C,0D72FH
D42D  C9          RET

*************** !DIR
D42E  CD73CD      CALL   0CD73H       ;sauvegarder pile
D431  0600        LD     B,00H
D433  B7          OR     A            ;est-ce que des paramètres suivent?
D434  2806        JR     Z,0D43CH     ;=> pas d'autre paramètre
D436  CDC2CD      CALL   0CDC2H       ;un param.est OK, sinon 'Bad command'
D439  CDC7CD      CALL   0CDC7H       ;b:=longu.chaîne, hl:=adresse chaîne
D43C  CDA6DA      CALL   0DAA6H       ;convertir en nom fichier correct
D43F  CD14CE      CALL   0CE14H       ;header param.disc dans hl, Login
                                       s'il y a lieu
D442  CDD0DB      CALL   0DBD0H       ;sortir 'Drive #: user #'
D445  3E0C        LD     A,0CH        ;longueur d'un nom fichier avec '.'
D447  CD72D4      CALL   0D472H       ;détermine nb entrées par ligne écran
D44A  65          LD     H,L
D44B  E5          PUSH   HL
D44C  CD83D6      CALL   0D683H       ;Nb entrées dir & tab affect sur 0
D44F  CD98D6      CALL   0D698H       ;chercher nom fichier et déterminer
                                       affectation
D452  301A        JR     NC,0D46EH    ;=> pas d'autre entrée
D454  CDDFD9      CALL   0D9DFH       ;teste, si entrée porte attribut SYS
D457  38F6        JR     C,0D44FH     ;=> 2ème car. ext. >7f, attribut SYS
D459  E3          EX     (SP),HL      ;entrées comptées sur pile
D45A  C5          PUSH   BC
D45B  7C          LD     A,H          ;h:= entrées/ligne
D45C  BD          CP     L            ;l:=encore entrées à sortir dns ligne
D45D  C4C4DB      CALL   NZ,0DBC4H    ;=> sortir trois espaces
D460  CCE9CA      CALL   Z,0CAE9H     ;=> sortir 'CR/LF'
D463  CDC8DB      CALL   0DBC8H       ;sortir nom de fichier, de pointe sur
                                      nom de fichier
D466  2D          DEC    L            ;entrées affichées dns ligne actuelle
D467  2001        JR     NZ,0D46AH    ;=> ligne pas encore pleine
```

```
D469  6C       LD    L,H        ;ligne pleine, nbre sur valeur
D46A  C1       POP   BC         ;de départ
D46B  E3       EX    (SP),HL    ;entrées comptées dans hl
D46C  18E1     JR    0D44FH     ;prochaine entrée

*************** fin !DIR
D46E  E1       POP   HL
D46F  C371D5   JP    0D571H     ;déterminer nbre blocs libres, sortir

*************** déterminer nombre entrées/ligne pour IDIR et CAT
D472  C603     ADD   A,03H      ;longueur entrée plus trois espaces
D474  67       LD    H,A
D475  D5       PUSH  DE
D476  E5       PUSH  HL
D477  CD69BB   CALL  0BB69H     ;TXT GET WINDOW
D47A  7A       LD    A,D        ;colonne droite de fenêtre actuelle
D47B  E1       POP   HL
D47C  D1       POP   DE
D47D  C604     ADD   A,04H
D47F  2E00     LD    L,00H      ;initialiser compteur entrées/ligne
D481  2C       INC   L          ;nombre entrées/ligne
D482  94       SUB   H          ;longueur d'une entrée DIR
D483  30FC     JR    NC,0D481H
D485  2D       DEC   L          ;nombre entrées DIR/ligne, correction
D486  C0       RET   NZ         ;si nombre=0, alors pas de formatage
D487  2E01     LD    L,01H      ;fixer nombre sur 1
D489  C9       RET

*************** IERA
D48A  CD73CD   CALL  0CD73H     ;sauvegarder pile
D48D  CDC2CD   CALL  0CDC2H     ;1 paramètre suit,sinon 'Bad command'
D490  CDC7CD   CALL  0CDC7H     ;adresse nom fichier à suppr. dans hl
D493  CD8DDA   CALL  0DA8DH     ;créer nom de fichier pour DOS
D496  CD14CE   CALL  0CE14H     ;header param.disc dans hl, login
                                 s'il y a lieu
D499  CD83D6   CALL  0D683H     ;nbre entrées Dir & tab affect sur 0
D49C  CD98D6   CALL  0D698H     ;cherche nom fichier et déterm.affect
D49F  306B     JR    NC,0D50CH  ;=>annonce fichier pas trouvé,interr.
D4A1  CDB1D4   CALL  0D4B1H
D4A4  CD98D6   CALL  0D698H     ;cherche nom fichier et déterm.affect
D4A7  38F8     JR    C,0D4A1H
D4A9  C9       RET
```

```
***************
D4AA  CDB1D4    CALL    OD4B1H
D4AD  D2B8CD    JP      NC,OCDB8H
D4B0  C9        RET

***************
D4B1  CDD9D9    CALL    OD9D9H      ;teste, si fichier READ ONLY
D4B4  3F        CCF
D4B5  3E0A      LD      A,OAH       ;message système '"nom fichier" is
                                     read only'
D4B7  D2CADB    JP      NC,ODBCAH   ;=> sortir message, interruption
D4BA  AF        XOR     A
D4BB  CD3CD8    CALL    OD83CH      ;libérer blocs dans table affectation
D4BE  3EE5      LD      A,OE5H      ;entrer marque fichier supprimé
D4C0  12        LD      (DE),A      ;dans nom de fichier
D4C1  C37AD9    JP      OD97AH

*************** IREN
D4C4  CD73CD    CALL    OCD73H      ;sauvegarder pile
D4C7  CDC1CD    CALL    OCDC1H      ;2 param.suivent, sinon interruption
D4CA  CDC7CD    CALL    OCDC7H      ;premier param., nouveau nom, dans hl
D4CD  CD5BDA    CALL    ODA5BH      ;organise nom de fichier pour DOS
D4D0  C5        PUSH    BC
D4D1  CDC7CD    CALL    OCDC7H      ;second param.,ancien nom, dans hl
D4D4  CD60DA    CALL    ODA60H      ;organise nom de fichier pour DOS
D4D7  E1        POP     HL
D4D8  0A        LD      A,(BC)
D4D9  BE        CP      (HL)
D4DA  C2AFCD    JP      NZ,OCDAFH   ;sortir Bad command, interr. instr.
D4DD  CD14CE    CALL    OCE14H      ;header param.disc dans hl, login
D4E0  23        INC     HL          ;s'il y a lieu
D4E1  E5        PUSH    HL
D4E2  CD44D6    CALL    OD644H      ;teste, si nouveau nom fichier existe
D4E5  E1        POP     HL          ;déjà
D4E6  C5        PUSH    BC
D4E7  44        LD      B,H
D4E8  4D        LD      C,L
D4E9  CD83D6    CALL    OD683H      ;nbre entrées Dir & tab affect sur 0
D4EC  CD98D6    CALL    OD698H      ;cherche nom fichier, déterm.affect.
D4EF  301B      JR      NC,OD50CH   ;=> fichier pas trouvé
D4F1  CDD9D9    CALL    OD9D9H      ;teste, si fichier READ ONLY
D4F4  DA8DD3    JP      C,OD38DH    ;=>fichier READ ONLY, pas changer nom
```

```
D4F7  E3        EX      (SP),HL
D4F8  E5        PUSH    HL
D4F9  C5        PUSH    BC
D4FA  010C00    LD      BC,000CH     ;longueur nouveau nom de fichier
D4FD  EDB0      LDIR                 ;remplacer ancien nom de fichier
D4FF  C1        POP     BC
D500  E1        POP     HL
D501  E3        EX      (SP),HL
D502  CD7AD9    CALL    0D97AH
D505  CD98D6    CALL    0D698H       ;chercher nom de fichier et
D508  38E7      JR      C,0D4F1H     ;déterminer affectation
D50A  E1        POP     HL
D50B  C9        RET

*************** fichier pas trouvé, interruption
D50C  50        LD      D,B          ;adr. nom fichier pour sortie dans de
D50D  59        LD      E,C
D50E  3E06      LD      A,06H        ;sortir message système 6, file not
D510  C3B1CD    JP      0CDB1H       ;found, interrompre instruction

*************** CATALOG
D513  CD73CD    CALL    0CD73H       ;ranger pointeur de pile
D516  D5        PUSH    DE           ;adresse buffer user
D517  DDE1      POP     IX           ;transférer dans ix
D519  010008    LD      BC,0800H     ;longueur du buffer user
D51C  CDAFCA    CALL    0CAAFH       ;vide buffer user (de) à (de+bc)
D51F  CD86DA    CALL    0DA86H
D522  CD14CE    CALL    0CE14H       ;header param.disc dans hl, login
                                      s'il y a lieu
D525  CDD0DB    CALL    0DBD0H       ;sortie 'Drive #: user #'
D528  AF        XOR     A
D529  F5        PUSH    AF
D52A  CD83D6    CALL    0D683H       ;nbre entrées Dir & tab affect sur 0
D52D  CD98D6    CALL    0D698H       ;cherche nom fichier déterm.affect.
D530  300C      JR      NC,0D53EH
D532  CDDFD9    CALL    0D9DFH       ;teste, si fichier porte attribut SYS
D535  38F6      JR      C,0D52DH     ;=> est fichier SYS, ne pas sortir
D537  E3        EX      (SP),HL
D538  CDAAD5    CALL    0D5AAH       ;une entrée dans buffer user
D53B  E3        EX      (SP),HL
D53C  38EF      JR      C,0D52DH     ;prochaine entrée, donc encore une
D53E  3E11      LD      A,11H        ;longueur d'une entr.DIR sur moniteur
```

```
D540  CD72D4    CALL    0D472H      ;détermine nombre d'entrées/ligne
D543  55        LD      D,L         ;nombre dans d
D544  F1        POP     AF
D545  1E00      LD      E,00H
D547  1C        INC     E
D548  92        SUB     D
D549  30FC      JR      NC,0D547H
D54B  82        ADD     A,D
D54C  2001      JR      NZ,0D54FH
D54E  1D        DEC     E           ;nombre de lignes
D54F  DDE5      PUSH    IX          ;buffer user
D551  E1        POP     HL          ;dans hl
D552  4B        LD      C,E         ;lignes dans c
D553  42        LD      B,D         ;entrées/ligne dans b
D554  E5        PUSH    HL
D555  CD7AD5    CALL    0D57AH      ;sortie d'une entrée
D558  D5        PUSH    DE
D559  EB        EX      DE,HL
D55A  2600      LD      H,00H
D55C  CD3AD6    CALL    0D63AH      ;hl:= hl * 14 pour sortie alphab.
D55F  19        ADD     HL,DE       ;dans plusieurs colonnes
D560  D1        POP     DE
D561  10F2      DJNZ    0D555H      ;et sortir prochaine entrée
D563  E1        POP     HL
D564  D5        PUSH    DE
D565  110E00    LD      DE,000EH
D568  19        ADD     HL,DE
D569  D1        POP     DE
D56A  0D        DEC     C
D56B  2804      JR      Z,0D571H
D56D  7E        LD      A,(HL)
D56E  B7        OR      A
D56F  20E2      JR      NZ,0D553H
D571  CDC2D8    CALL    0D8C2H      ;détermine nombre blocs occupés
D574  3E03      LD      A,03H
D576  B7        OR      A           ;sortir message système 3, xxxK free
D577  C3EBCA    JP      0CAEBH

*************** sortie d'une entrée du buffer user
D57A  E7        RST     20H         ;RAM LAM, LD A,(HL) de buffer user
D57B  B7        OR      A           ;caractère un 0?
D57C  C8        RET     Z           ;alors pas d'entrée et =>
```

```
D57D  E5       PUSH   HL          ;sauver registres
D57E  D5       PUSH   DE
D57F  C5       PUSH   BC
D580  78       LD     A,B         ;entrées/ligne
D581  BA       CP     D           ;égale entrées sorties?
D582  C4C4DB   CALL   NZ,0DBC4H   ;si différent, sortir 3 espaces
D585  CCE9CA   CALL   Z,0CAE9H    ;sinon sortir 'CR/LF'
D588  EB       EX     DE,HL
D589  CDC8DB   CALL   0DBC8H      ;sortir nom de fichier
D58C  CDD9D9   CALL   0D9D9H      ;teste, si fichier READ ONLY
D58F  3E2A     LD     A,2AH       ;'*'
D591  3802     JR     C,0D595H    ;si R/O, sortir '*'
D593  3E20     LD     A,20H       ;'espace'
D595  CD5ABB   CALL   0BB5AH      ;TXT OUTPUT
D598  210C00   LD     HL,000CH
D59B  19       ADD    HL,DE
D59C  E7       RST    20H         ;RAM LAM, LD A,(HL) de Ram
D59D  5F       LD     E,A
D59E  23       INC    HL
D59F  E7       RST    20H         ;RAM LAM, LD A,(HL) de Ram
D5A0  57       LD     D,A
D5A1  3E02     LD     A,02H       ;message système 2, sortir
D5A3  CDEBCA   CALL   0CAEBH      ;3 espaces
D5A6  C1       POP    BC
D5A7  D1       POP    DE
D5A8  E1       POP    HL
D5A9  C9       RET

***************

D5AA  C5       PUSH   BC
D5AB  4C       LD     C,H
D5AC  0600     LD     B,00H
D5AE  DDE5     PUSH   IX          ;buffer user
D5B0  E1       POP    HL          ;dans hl
D5B1  E7       RST    20H         ;RAM LAM, LD A,(HL) de Ram
D5B2  B7       OR     A           ;teste si caractère 0
D5B3  2850     JR     Z,0D605H
D5B5  04       INC    B
D5B6  CD23D6   CALL   0D623H
D5B9  280F     JR     Z,0D5CAH
D5BB  3026     JR     NC,0D5E3H
D5BD  D5       PUSH   DE
```

```
D5BE  110E00    LD      DE,000EH
D5C1  19        ADD     HL,DE
D5C2  D1        POP     DE
D5C3  78        LD      A,B
D5C4  FE92      CP      92H
D5C6  38E9      JR      C,0D5B1H
D5C8  1856      JR      0D620H

***************
D5CA  E5        PUSH    HL
D5CB  CDF2D8    CALL    0D8F2H        ;déterm.nbre blocs occupés du fichier
D5CE  E3        EX      (SP),HL
D5CF  110C00    LD      DE,000CH
D5D2  19        ADD     HL,DE
D5D3  E7        RST     20H           ;RAM LAM, LD A,(HL) de Ram
D5D4  5F        LD      E,A
D5D5  23        INC     HL
D5D6  E7        RST     20H           ;RAM LAM, LD A,(HL) de Ram
D5D7  57        LD      D,A
D5D8  2B        DEC     HL
D5D9  E3        EX      (SP),HL
D5DA  19        ADD     HL,DE
D5DB  EB        EX      DE,HL
D5DC  E1        POP     HL
D5DD  73        LD      (HL),E
D5DE  23        INC     HL
D5DF  72        LD      (HL),D
D5E0  37        SCF
D5E1  183D      JR      0D620H

***************
D5E3  79        LD      A,C
D5E4  FE92      CP      92H
D5E6  2838      JR      Z,0D620H
D5E8  E5        PUSH    HL
D5E9  D5        PUSH    DE
D5EA  C5        PUSH    BC
D5EB  EB        EX      DE,HL
D5EC  79        LD      A,C
D5ED  90        SUB     B
D5EE  3C        INC     A
D5EF  6F        LD      L,A
```

```
D5F0  2600     LD     H,00H
D5F2  CD3AD6   CALL   0D63AH
D5F5  44       LD     B,H
D5F6  4D       LD     C,L
D5F7  19       ADD    HL,DE
D5F8  2B       DEC    HL
D5F9  EB       EX     DE,HL
D5FA  210E00   LD     HL,000EH
D5FD  19       ADD    HL,DE
D5FE  EB       EX     DE,HL
D5FF  CD1EB9   CALL   0B91EH
D602  C1       POP    BC
D603  D1       POP    DE
D604  E1       POP    HL

***************

D605  0C       INC    C
D606  C5       PUSH   BC
D607  D5       PUSH   DE
D608  36FF     LD     (HL),0FFH   ;marque fin dans buffer user
D60A  23       INC    HL
D60B  13       INC    DE          ;de := pointeur d'enregistrement
D60C  EB       EX     DE,HL
D60D  010B00   LD     BC,000BH    ;longueur nom de fichier + extension
D610  CD1BB9   CALL   0B91BH      ;KL LDIR, d'enreg. dans buffer user
D613  EB       EX     DE,HL       ;pointeur d'enregistrement dans hl
D614  E3       EX     (SP),HL     ;début d'enregistrement dans hl
D615  EB       EX     DE,HL       ;début d'enregistrement dans de
D616  CDF2D8   CALL   0D8F2H      ;déterm.nbre blocs occupés du fichier
D619  EB       EX     DE,HL       ;nombre blocs dans de
D61A  E1       POP    HL
D61B  73       LD     (HL),E      ;entrer dans buffer user
D61C  23       INC    HL
D61D  72       LD     (HL),D
D61E  C1       POP    BC
D61F  37       SCF
D620  61       LD     H,C
D621  C1       POP    BC
D622  C9       RET

***************

D623  E5       PUSH   HL
```

```
D624  D5        PUSH   DE
D625  C5        PUSH   BC
D626  060B      LD     B,0BH
D628  13        INC    DE
D629  23        INC    HL
D62A  1A        LD     A,(DE)
D62B  E67F      AND    7FH
D62D  4F        LD     C,A
D62E  E7        RST    20H         ;RAM LAM, LD A,(HL) de Ram
D62F  E67F      AND    7FH
D631  B9        CP     C
D632  2002      JR     NZ,0D636H
D634  10F2      DJNZ   0D628H
D636  C1        POP    BC
D637  D1        POP    DE
D638  E1        POP    HL
D639  C9        RET

*************** hl:=hl*14
D63A  D5        PUSH   DE          ;ranger
D63B  54        LD     D,H
D63C  5D        LD     E,L
D63D  29        ADD    HL,HL       ;hl dernière valeur dans hl*2
D63E  19        ADD    HL,DE       ;hl dernière valeur dans hl*3
D63F  29        ADD    HL,HL       ;hl dernière valeur dans hl*6
D640  19        ADD    HL,DE       ;hl dernière valeur dans hl*7
D641  29        ADD    HL,HL       ;hl dernière valeur dans hl*14
D642  D1        POP    DE
D643  C9        RET

***************
D644  CD83D6    CALL   0D683H      ;nbre entrées Dir & tab affect sur 0
D647  CD98D6    CALL   0D698H      ;cherche nom fichier et déterm.affect
D64A  3025      JR     NC,0D671H
D64C  3E05      LD     A,05H       ;sortir message système 5, file
D64E  C3B1CD    JP     0CDB1H      ;already exists, interrompre instr.

*************** chercher nom de fichier dans le directory
D651  CD83D6    CALL   0D683H      ;nbre entrées dir et tab affect sur 0
D654  CD98D6    CALL   0D698H      ;cherche nom fichier et déterm.affect
D657  3018      JR     NC,0D671H   ;=> pas trouvé
D659  E5        PUSH   HL          ;numéro entrée Dir trouvée
```

```
D65A  210900    LD     HL,0009H
D65D  CD9FCA    CALL   0CA9FH        ;hl=hl+iy, adresse FCN OPENIN
D660  EB        EX     DE,HL         ;dans de
D661  CDDFDB    CALL   0DBDFH        ;32 octets,entrée DIR dans FCB OPENIN
D664  E1        POP    HL            ;numéro de l'entrée DIR trouvée
D665  FD7E05    LD     A,(IY+05H)    ;fichier actif sur ce lecteur?
D668  B7        OR     A
D669  37        SCF
D66A  C0        RET    NZ            ;=> fichier est actif
D66B  CDA2D6    CALL   0D6A2H        ;lire directory jusqu'à fin, déterm.
D66E  38FB      JR     C,0D66BH      ;nombres fichiers et blocs occupés
D670  37        SCF                  ;Directory lu
D671  FD3605FF  LD     (IY+05H),0FF;marquer fichier sur ce lecteur comme
D675  C9        RET                  ;actif

****** cherche nom de fichier dans directory, détermine affectation blocs
D676  CD83D6    CALL   0D683H        ;nbre entrées Dir & tab affect sur 0
D679  CD98D6    CALL   0D698H        ;cherche nom fichier et déterm.affect
D67C  30F3      JR     NC,0D671H     ;=> pas trouvé
D67E  CDAAD4    CALL   0D4AAH        ;
D681  18F6      JR     0D679H

*************** déterminer affectation blocs disque
D683  C5        PUSH   BC            ;=> nom de fichier
D684  CD1FC5    CALL   0C51FH        ;chercher piste 0
D687  C1        POP    BC
D688  21FFFF    LD     HL,0FFFFH
D68B  FD7E05    LD     A,(IY+05H)    ;fichier sur ce lecteur actif?
D68E  B7        OR     A
D68F  C0        RET    NZ            ;=> fichier est ouvert
D690  E5        PUSH   HL
D691  CD14D8    CALL   0D814H        ;tab affect à 0, occuper blocs Dir
D694  E1        POP    HL
D695  C3A8D9    JP     0D9A8H

************ chercher nom de fichier, affectation blocs + nombre fichiers
D698  CDA2D6    CALL   0D6A2H        ;compte nombre blocs dans table
                                      d'affectation, fichiers
D69B  D0        RET    NC            ;=> pas d'autre entrée
D69C  CDD8D7    CALL   0D7D8H        ;nom fichier indiqué= entrée Dir?
D69F  30F7      JR     NC,0D698H     ;=> différent, chercher et déterminer
                                     affectation
```

```
D6A1  C9        RET                 ;trouvé même nom de fichier

*************** déterminer nombre de fichiers sur disque
D6A2  23        INC     HL          ;compteur d'entrées
D6A3  FD7E05    LD      A,(IY+05H)  ;OPEN actif sur ce lecteur?
D6A6  B7        OR      A
D6A7  2011      JR      NZ,0D6BAH   ;=> Open est actif
D6A9  CD1CD9    CALL    0D91CH      ;enregistrement dans buffer, pointeur
                                     d'enregistrement => entrée Dir
D6AC  D0        RET     NC          ;=> pas d'autre entrée
D6AD  1A        LD      A,(DE)      ;de pointeur sur entrée Dir
D6AE  FEE5      CP      0E5H        ;marque fichier supprimé
D6B0  37        SCF
D6B1  C8        RET     Z           ;=> si entrée supprimée
D6B2  CDA8D9    CALL    0D9A8H      ;sinon augmenter nombre d'entrées
D6B5  3EFF      LD      A,0FFH
D6B7  C33CD8    JP      0D83CH      ;détermine nombre de blocs occupés
```

```
*************** entrée D6A2, si Open actif
D6BA  CDB8D9    CALL   0D9B8H      ;teste, si pos. entrée dir ok
D6BD  D0        RET    NC
D6BE  C31CD9    JP     0D91CH

*************** nombre caractères dans le fichier dans hl
D6C1  212100    LD     HL,0021H
D6C4  19        ADD    HL,DE
D6C5  C3F9DB    JP     0DBF9H      ;ld hl,(hl)

***************
D6C8  212300    LD     HL,0023H
D6CB  19        ADD    HL,DE
D6CC  7E        LD     A,(HL)
D6CD  B7        OR     A
D6CE  C0        RET    NZ
D6CF  CDC1D6    CALL   0D6C1H
D6D2  7C        LD     A,H
D6D3  1F        RRA
D6D4  1F        RRA
D6D5  1F        RRA
D6D6  1F        RRA
D6D7  E60F      AND    0FH
D6D9  47        LD     B,A
D6DA  29        ADD    HL,HL
D6DB  7C        LD     A,H
D6DC  E61F      AND    1FH
D6DE  4F        LD     C,A
D6DF  C5        PUSH   BC
D6E0  210F00    LD     HL,000FH
D6E3  19        ADD    HL,DE
D6E4  7E        LD     A,(HL)
D6E5  A8        XOR    B
D6E6  200F      JR     NZ,0D6F7H
D6E8  3E04      LD     A,04H
D6EA  CD54DA    CALL   0DA54H      ;masque extension dans accu
D6ED  2F        CPL
D6EE  47        LD     B,A
D6EF  2B        DEC    HL
D6F0  2B        DEC    HL
D6F1  7E        LD     A,(HL)
D6F2  A9        XOR    C
```

```
D6F3  A0        AND    B
D6F4  2001      JR     NZ,0D6F7H
D6F6  37        SCF
D6F7  C1        POP    BC
D6F8  9F        SBC    A,A
D6F9  C9        RET

***************
D6FA  210D00    LD     HL,000DH
D6FD  19        ADD    HL,DE
D6FE  71        LD     (HL),C
D6FF  23        INC    HL
D700  23        INC    HL
D701  70        LD     (HL),B
D702  23        INC    HL
D703  EB        EX     DE,HL
D704  011100    LD     BC,0011H
D707  CDAFCA    CALL   0CAAFH      ;vide (de) à (de+bc)
D70A  EB        EX     DE,HL
D70B  C9        RET

***************
D70C  D5        PUSH   DE
D70D  CDC1D6    CALL   0D6C1H
D710  7C        LD     A,H
D711  E60F      AND    0FH
D713  67        LD     H,A
D714  E5        PUSH   HL
D715  211000    LD     HL,0010H
D718  19        ADD    HL,DE
D719  4E        LD     C,(HL)
D71A  0600      LD     B,00H
D71C  2B        DEC    HL
D71D  2B        DEC    HL
D71E  2B        DEC    HL
D71F  66        LD     H,(HL)
D720  68        LD     L,B
D721  3E01      LD     A,01H
D723  CDEBDB    CALL   0DBEBH
D726  09        ADD    HL,BC
D727  D1        POP    DE
D728  13        INC    DE
```

```
D729  CDF3DB    CALL    0DBF3H      ;hl = de? Carry si égal
D72C  3F        CCF
D72D  D1        POP     DE
D72E  C9        RET

***************
D72F  CDC1D6    CALL    0D6C1H
D732  3E03      LD      A,03H
D734  CD54DA    CALL    0DA54H      ;masque bloc dans accu
D737  A5        AND     L
D738  4F        LD      C,A
D739  3E02      LD      A,02H
D73B  CD54DA    CALL    0DA54H      ;Shift bloc dans accu
D73E  CDEBDB    CALL    0DBEBH
D741  3E06      LD      A,06H
D743  CD54DA    CALL    0DA54H      ;No bloc maxi, octet fort dans accu
D746  47        LD      B,A
D747  B7        OR      A
D748  7D        LD      A,L
D749  211100    LD      HL,0011H
D74C  19        ADD     HL,DE
D74D  280E      JR      Z,0D75DH
D74F  E607      AND     07H
D751  87        ADD     A,A
D752  85        ADD     A,L
D753  6F        LD      L,A
D754  8C        ADC     A,H
D755  95        SUB     L
D756  67        LD      H,A
D757  E5        PUSH    HL
D758  CDF9DB    CALL    0DBF9H      ;ld hl,(hl)
D75B  180B      JR      0D768H

***************
D75D  E60F      AND     0FH
D75F  85        ADD     A,L
D760  6F        LD      L,A
D761  8C        ADC     A,H
D762  95        SUB     L
D763  67        LD      H,A
D764  E5        PUSH    HL
D765  6E        LD      L,(HL)
```

```
D766  2600      LD     H,00H
D768  7C        LD     A,H
D769  B5        OR     L
D76A  280F      JR     Z,0D77BH
D76C  F1        POP    AF
D76D  3E02      LD     A,02H
D76F  CD54DA    CALL   0DA54H       ;Shift bloc dans accu
D772  29        ADD    HL,HL
D773  3D        DEC    A
D774  20FC      JR     NZ,0D772H
D776  79        LD     A,C
D777  B5        OR     L
D778  6F        LD     L,A
D779  37        SCF
D77A  C9        RET

***************
D77B  E1        POP    HL
D77C  C9        RET

*************** déterminer nombre enregistrements dans FCB
D77D  211000    LD     HL,0010H
D780  19        ADD    HL,DE
D781  7E        LD     A,(HL)
D782  34        INC    (HL)
D783  B7        OR     A
D784  F0        RET    P
D785  3601      LD     (HL),01H
D787  2B        DEC    HL
D788  2B        DEC    HL
D789  2B        DEC    HL
D78A  34        INC    (HL)
D78B  C9        RET

*************** occuper entrée directory libre
D78C  D5        PUSH   DE
D78D  D5        PUSH   DE
D78E  CDBBD7    CALL   0D7BBH       ;chercher entrée libre
D791  E3        EX     (SP),HL      ;adr. entrée sur pile,hl=adr.nom fich
D792  23        INC    HL           ;No lecteur pas nécessaire
D793  CDDFDB    CALL   0DBDFH       ;écrire nom fichier et affect bloc
                                     dans enregistrement
```

```
D796  E1        POP    HL
D797  CD7AD9    CALL   0D97AH      ;nom de fichier et affectation blocs
D79A  D1        POP    DE          ;dans enregistrement Dir
D79B  C9        RET

*************** fixer nombre caractères dans le fichier sur 0
D79C  212100    LD     HL,0021H
D79F  19        ADD    HL,DE
D7A0  AF        XOR    A
D7A1  77        LD     (HL),A
D7A2  23        INC    HL
D7A3  77        LD     (HL),A
D7A4  23        INC    HL
D7A5  77        LD     (HL),A
D7A6  C9        RET

*************** augmenter de 1 nombre caractères dans le fichier
D7A7  212100    LD     HL,0021H
D7AA  19        ADD    HL,DE
D7AB  34        INC    (HL)
D7AC  C0        RET    NZ
D7AD  23        INC    HL
D7AE  34        INC    (HL)
D7AF  C0        RET    NZ
D7B0  23        INC    HL
D7B1  34        INC    (HL)
D7B2  C9        RET

***************
D7B3  CD83D6    CALL   0D683H      ;nbre entrées Dir & tab affect sur 0
D7B6  CD98D6    CALL   0D698H      ;cherche nom fichier et déterm.affect
D7B9  1811      JR     0D7CCH

*************** chercher entrée directory libre
D7BB  21FFFF    LD     HL,0FFFFH
D7BE  23        INC    HL
D7BF  CD1CD9    CALL   0D91CH      ;teste si place dans directory
D7C2  3E07      LD     A,07H       ;sortir mess.système 7,directory full
D7C4  D2B1CD    JP     NC,0CDB1H   ;interrompre instruction si pas place
D7C7  1A        LD     A,(DE)      ;(de)= pointeur enreg.dans enreg. Dir
D7C8  FEE5      CP     0E5H        ;entrée supprimée?
D7CA  20F2      JR     NZ,0D7BEH   ;=> pas libre, entrée suivante
```

```
D7CC  F5          PUSH    AF
D7CD  FD7E05      LD      A,(IY+05H)  ;OPEN actif sur ce lecteur?
D7D0  B7          OR      A
D7D1  3E09        LD      A,09H
D7D3  CAB8CD      JP      Z,0CDB8H    ;erreur, interrompre instruction
D7D6  F1          POP     AF
D7D7  C9          RET

*************** cherche dans le directory le nom de fichier indiqué en bc
D7D8  C5          PUSH    BC          ;(bc) => nom fichier du fichier voulu
D7D9  D5          PUSH    DE          ;(de) => pointeur d'enregistrement
D7DA  E5          PUSH    HL
D7DB  60          LD      H,B         ;adresse nom fichier indiqué dans hl
D7DC  69          LD      L,C
D7DD  1A          LD      A,(DE)      ;premier caractère entrée Dir,No user
D7DE  AE          XOR     (HL)        ;égal à numéro user indiqué?
D7DF  202D        JR      NZ,0D80EH   ;si non, alors =>
D7E1  23          INC     HL
D7E2  13          INC     DE
D7E3  060B        LD      B,0BH       ;longueur nom de fichier + extension
D7E5  7E          LD      A,(HL)
D7E6  FE3F        CP      3FH         ;joker '?' dans nom fichier indiqué?
D7E8  2806        JR      Z,0D7F0H    ;=> ignorer caractère dans entrée Dir
D7EA  1A          LD      A,(DE)      ;caractère d'entrée Dir
DEB  AE          XOR     (HL)        ;comparer avec caract. du nom fourni
D7EC  E67F        AND     7FH
D7EE  201E        JR      NZ,0D80EH   ;=> les noms sont différents
D7F0  23          INC     HL          ;sinon prochain caract. nom indiqué
D7F1  13          INC     DE          ;prochain caractère entrée Dir
D7F2  10F1        DJNZ    0D7E5H      ;tester
D7F4  7E          LD      A,(HL)
D7F5  3C          INC     A
D7F6  280C        JR      Z,0D804H
D7F8  3E04        LD      A,04H
D7FA  CD54DA      CALL    0DA54H      ;masque extension dans accu
D7FD  2F          CPL
D7FE  47          LD      B,A
D7FF  1A          LD      A,(DE)
D800  AE          XOR     (HL)
D801  A0          AND     B
D802  200A        JR      NZ,0D80EH
D804  23          INC     HL
```

```
D805  13          INC     DE
D806  23          INC     HL
D807  13          INC     DE
D808  7E          LD      A,(HL)
D809  3C          INC     A
D80A  2802        JR      Z,0D80EH
D80C  1A          LD      A,(DE)
D80D  AE          XOR     (HL)
D80E  E1          POP     HL
D80F  D1          POP     DE
D810  C1          POP     BC
D811  C0          RET     NZ             ;fichier pas trouvé
D812  37          SCF                    ;marque OK
D813  C9          RET

*************** vide table affectation, inscrit blocs Dir
D814  3E05        LD      A,05H
D816  CD45DA      CALL    0DA45H         ;numéro bloc maxi dans hl
D819  3E03        LD      A,03H
D81B  CDEBDB      CALL    0DBEBH         ;divise numéro bloc par 8
D81E  23          INC     HL             ;correction, 22 octets pour table
D81F  EB          EX      DE,HL          ;d'affectation
D820  3E0E        LD      A,0EH
D822  CD3FDA      CALL    0DA3FH         ;(hl)=> début table act.d'affectation
D825  3600        LD      (HL),00H       ;8 blocs = 1 octet comme marque libre
D827  23          INC     HL             ;octet suivant
D828  1B          DEC     DE             ;diminuer nombre
D829  7A          LD      A,D            ;les 22 octets tous annulés?
D82A  B3          OR      E
D82B  20F8        JR      NZ,0D825H      ;=> pas encore tous annulés
D82D  3E09        LD      A,09H
D82F  CD45DA      CALL    0DA45H         ;taille en blocs de directory dans hl
D832  EB          EX      DE,HL
D833  3E0E        LD      A,0EH
D835  CD3FDA      CALL    0DA3FH         ;(hl)=> début table act.d'affectation
D838  73          LD      (HL),E         ;inscrire blocs Dir comme occupés
D839  23          INC     HL
D83A  72          LD      (HL),D
D83B  C9          RET

*************** occupe blocs de cette entrée dans table d'affectation
D83C  E5          PUSH    HL             ;nombre entrées Dir testées
```

```
D83D  D5      PUSH  DE         ;pointeur sur début entrée dns enreg.
D83E  C5      PUSH  BC
D83F  4F      LD    C,A
D840  211000  LD    HL,0010H   ;décalage par rapport affectat. blocs
D843  19      ADD   HL,DE
D844  0610    LD    B,10H      ;nombre d'entrées d'affectation par
D846  5E      LD    E,(HL)     ;entrée Dir
D847  23      INC   HL
D848  3E06    LD    A,06H
D84A  CD54DA  CALL  0DA54H     ;No bloc maxi, octet fort dans accu
D84D  B7      OR    A          ;si 0 alors entrées sur 1 octet
D84E  2803    JR    Z,0D853H   ;=> entrées sur un octet
D850  05      DEC   B
D851  7E      LD    A,(HL)
D852  23      INC   HL
D853  57      LD    D,A
D854  B3      OR    E
D855  280E    JR    Z,0D865H   ;pas d'autres blocs occupés
D857  E5      PUSH  HL
D858  3E05    LD    A,05H
D85A  CD45DA  CALL  0DA45H     ;numéro bloc maxi dans hl
D85D  7D      LD    A,L
D85E  93      SUB   E          ;numéro bloc valable?
D85F  7C      LD    A,H
D860  9A      SBC   A,D
D861  D46CD8  CALL  NC,0D86CH  ;valable=> occuper/libérer
D864  E1      POP   HL         ;dans table d'affectation
D865  10DF    DJNZ  0D846H
D867  C1      POP   BC
D868  D1      POP   DE
D869  E1      POP   HL
D86A  37      SCF
D86B  C9      RET

*************** occuper/libérer No bloc dans position bit table d'affect.
D86C  C5      PUSH  BC
D86D  D5      PUSH  DE         ;numéro bloc occupé
D86E  D5      PUSH  DE
D86F  EB      EX    DE,HL
D870  3E03    LD    A,03H
D872  CDEBDB  CALL  0DBEBH     ;diviser No bloc par 8
D875  EB      EX    DE,HL      ;résultat dns de,octet dns tab affect
```

```
D876  3E0E     LD     A,0EH
D878  CD3FDA   CALL   0DA3FH      ;(hl)=> début table affectation act.
D87B  19       ADD    HL,DE       ;sauter nombre d'octets nécessaire
D87C  D1       POP    DE          ;numéro de bloc occupé dans de
D87D  7B       LD     A,E
D87E  E607     AND    07H
D880  5F       LD     E,A
D881  3E01     LD     A,01H
D883  1C       INC    E
D884  0F       RRCA               ;déterminer position bit dans table
D885  1D       DEC    E           ;d'affectation
D886  20FC     JR     NZ,0D884H
D888  47       LD     B,A
D889  A1       AND    C
D88A  4F       LD     C,A
D88B  78       LD     A,B
D88C  2F       CPL
D88D  A6       AND    (HL)
D88E  B1       OR     C           ;ORer modèle bits act. avec bit bloc
D88F  77       LD     (HL),A      ;et sauvegarder dans table d'affect.
D890  D1       POP    DE
D891  C1       POP    BC
D892  C9       RET

*************** cherche bloc libre dans table d'affectation
D893  C5       PUSH   BC
D894  D5       PUSH   DE
D895  3E05     LD     A,05H
D897  CD45DA   CALL   0DA45H      ;numéro bloc maxi dans hl
D89A  EB       EX     DE,HL
D89B  3E0E     LD     A,0EH
D89D  CD3FDA   CALL   0DA3FH      ;(hi)=> début act. table d'affect.
D8A0  018008   LD     BC,0880H    ;b:= compteur de bits, c:=modèle bits
D8A3  7E       LD     A,(HL)      ;entrée de table affect. dans accu
D8A4  A1       AND    C           ;modèle bits pour bloc
D8A5  280C     JR     Z,0D8B3H    ;=> trouvé bloc libre
D8A7  0F       RRCA               ;tester bit suivant
D8A8  4F       LD     C,A
D8A9  7A       LD     A,D         ;de:= nombre blocs encore à tester
D8AA  B3       OR     E           ;plus d'autres blocs?
D8AB  2812     JR     Z,0D8BFH    ;=> trouvé aucun bloc libre,Disc Full
D8AD  1B       DEC    DE          ;diminuer nombre blocs à tester
```

```
D8AE  10F3      DJNZ    0D8A3H      ;boucle bits sur 8 bits
D8B0  23        INC     HL          ;prochain octet dans table d'affect.
D8B1  18ED      JR      0D8A0H      ;continuer à tester

* détermine No bloc d'après position bit, occupe dans table d'affectation
D8B3  7E        LD      A,(HL)      ;modèle bits de tab affect dans accu
D8B4  B1        OR      C           ;occuper bloc libre trouvé
D8B5  77        LD      (HL),A      ;et placer dans table d'affectation
D8B6  3E05      LD      A,05H
D8B8  CD45DA    CALL    0DA45H      ;numéro de bloc maxi dans hl
D8BB  B7        OR      A
D8BC  ED52      SBC     HL,DE       ;calculer numéro de bloc, dans hl
D8BE  37        SCF                 ;marque que trouvé bloc libre
D8BF  D1        POP     DE
D8C0  C1        POP     BC
D8C1  C9        RET

***** nombre bloc occupés de table d'affectation pour affichage directory
D8C2  C5        PUSH    BC
D8C3  E5        PUSH    HL
D8C4  210000    LD      HL,0000H    ;hl est compteur de blocs
D8C7  E5        PUSH    HL          ;sur pile
D8C8  3E05      LD      A,05H
D8CA  CD45DA    CALL    0DA45H      ;numéro de bloc maxi dans hl
D8CD  EB        EX      DE,HL       ;et dans de
D8CE  3E0E      LD      A,0EH
D8D0  CD3FDA    CALL    0DA3FH      ;(hl) => début actuelle ALV
D8D3  018008    LD      BC,0880H    ;b:=compteur de bits, c:= modèle bits
D8D6  7E        LD      A,(HL)      ;octet de tab d'affect. dans accu
D8D7  A1        AND     C           ;position de bit marquée occupée?
D8D8  2003      JR      NZ,0D8DDH   ;=> pas occupé
D8DA  E3        EX      (SP),HL     ;compteur dans hl
D8DB  23        INC     HL          ;augmenter compteur et
D8DC  E3        EX      (SP),HL     ;de nouveau sur la pile
D8DD  79        LD      A,C         ;modèle bits dans accu
D8DE  0F        RRCA                ;amener modèle bits suivant par rotat
D8DF  4F        LD      C,A         ;et ranger à nouveau dans c
D8E0  7A        LD      A,D         ;de:= nombre blocs encore à tester
D8E1  B3        OR      E           ;nombre déjà nul?
D8E2  2806      JR      Z,0D8EAH    ;=> plus d'autre bloc à tester
D8E4  1B        DEC     DE          ;diminuer nombre de blocs à tester
D8E5  10EF      DJNZ    0D8D6H      ;boucle de bits
```

```
D8E7  23        INC    HL          ;prochain octet de table d'affect.
D8E8  18E9      JR     0D8D3H      ;et continuer à tester

*************** détermine nombre effectif blocs occupés
D8EA  E1        POP    HL          ;retire comptr blocs occupés de pile
D8EB  CD10D9    CALL   0D910H      ;déterm.affectat.effect.avec BSH & hl
D8EE  EB        EX     DE,HL       ;résultat dans de
D8EF  E1        POP    HL
D8F0  C1        POP    BC
D8F1  C9        RET

*************** détermine nombre de blocs occupés par le fichier
D8F2  D5        PUSH   DE          ;début d'une entrée Dir dns enreg Dir
D8F3  211000    LD     HL,0010H    ;entrer décal.par rapport affect bloc
D8F6  19        ADD    HL,DE
D8F7  110010    LD     DE,1000H    ;d:= octets table d'affectation
D8FA  3E06      LD     A,06H       ;e:= compteur blocs occupés
D8FC  CD54DA    CALL   0DA54H      ;No bloc maxi, octet fort dans accu
D8FF  B7        OR     A           ;si 0,que valeurs 1 oct.ds tab affect
D900  7E        LD     A,(HL)      ;un octet de table d'affectation
D901  23        INC    HL          ;augmenter pointeur dans table affect
D902  2803      JR     Z,0D907H    ;octet fort No bloc maxi=0?, alors ne
                                   tester qu'un octet
D904  B6        OR     (HL)        ;sinon tester octet faible
D905  15        DEC    D
D906  23        INC    HL
D907  B7        OR     A           ;tester octet table affect. si 0
D908  2801      JR     Z,0D90BH    ;aucun bloc occupé
D90A  1C        INC    E           ;augmenter compteur blocs occupés
D90B  15        DEC    D           ;diminuer nombre
D90C  20EC      JR     NZ,0D8FAH   ;encore octets dans la table?
D90E  EB        EX     DE,HL       ;nombre dans hl
D90F  D1        POP    DE
D910  3E02      LD     A,02H
D912  CD54DA    CALL   0DA54H      ;Shift bloc dans accu
D915  3D        DEC    A
D916  3D        DEC    A
D917  3D        DEC    A
D918  C8        RET    Z
D919  29        ADD    HL,HL
D91A  18FB      JR     0D917H
```

```
***************
D91C  E5       PUSH   HL
D91D  C5       PUSH   BC          ;bc => nom fichier temporaire
D91E  7D       LD     A,L         ;nombre entrées testées ds l'enreg.
D91F  E603     AND    03H         ;4 sont possibles, 0-3
D921  2011     JR     NZ,0D934H
D923  EB       EX     DE,HL       ;nombre entrées dans de
D924  3E07     LD     A,07H
D926  CD45DA   CALL   0DA45H      ;entrées Dir maxi dans hl
D929  CDF3DB   CALL   0DBF3H      ;hl = de? testé toutes les entrées
D92C  3F       CCF                ;possibles
D92D  EB       EX     DE,HL
D92E  3015     JR     NC,0D945H   ;toutes lues =>
D930  CD48D9   CALL   0D948H      ;nouvel enreg. dans buffer enreg.
D933  AF       XOR    A
D934  47       LD     B,A
D935  3E08     LD     A,08H
D937  CD3FDA   CALL   0DA3FH      ;(hl)=>buffer enreg.,à nouv sur début
D93A  112000   LD     DE,0020H    ;longueur d'une entrée dans le buffer
D93D  04       INC    B           ;(32 octets)
D93E  1801     JR     0D941H

***************
D940  19       ADD    HL,DE       ;pointeur d'enregistrement sur
D941  10FD     DJNZ   0D940H      ;prochain début Dir
D943  EB       EX     DE,HL       ;pointeur d'enregistrement dans de
D944  37       SCF
D945  C1       POP    BC
```

```
D946  E1        POP     HL
D947  C9        RET

***************
D948  3E02      LD      A,02H
D94A  CDEBDB    CALL    0DBEBH      ;nombre d'entrées testées/4
D94D  EB        EX      DE,HL       ;résultat nbre enreg. testés dans de
D94E  3E08      LD      A,08H
D950  CD3FDA    CALL    0DA3FH      ;(hl) => numéro d'enregistrement
D953  CDE8D9    CALL    0D9E8H      ;amène enr., No ds hl, ds buffer enr.
D956  3E0B      LD      A,0BH
D958  CD45DA    CALL    0DA45H      ;nbre entrées DIR à tester dans hl
D95B  EB        EX      DE,HL
D95C  CDF3DB    CALL    0DBF3H      ;hl=de,testé toutes entrés possibles?
D95F  EB        EX      DE,HL
D960  D0        RET     NC          ;=> lu toutes les entrées
D961  3E0C      LD      A,0CH
D963  CD3FDA    CALL    0DA3FH      ;(hl) => bloc valeur de contrôle
D966  19        ADD     HL,DE
D967  CDC8D9    CALL    0D9C8H      ;calcul valeur contrôle sur l'enreg.
D96A  BE        CP      (HL)        ;égale valeur contrôle sauvegardée?
D96B  C8        RET     Z           ;=> sont égales
D96C  F5        PUSH    AF
D96D  EB        EX      DE,HL
D96E  29        ADD     HL,HL
D96F  29        ADD     HL,HL
D970  CDB8D9    CALL    0D9B8H
D973  EB        EX      DE,HL
D974  D1        POP     DE
D975  DA33D2    JP      C,0D233H    ;erreur
D978  72        LD      (HL),D
D979  C9        RET

***************
D97A  E5        PUSH    HL          ;numéro de l'entrée Dir occupée
D97B  C5        PUSH    BC          ;adresse buffer
D97C  3E02      LD      A,02H
D97E  CDEBDB    CALL    0DBEBH      ;divise hl/4
D981  EB        EX      DE,HL       ;résultat No enregistrement dans de
D982  3E08      LD      A,08H
D984  CD3FDA    CALL    0DA3FH      ;(hl) => buffer d'enregistrement
D987  0E01      LD      C,01H
```

```
D989  CDF3D9    CALL   0D9F3H      ;enregistrement Dir dans buffer
D98C  3E0B      LD     A,0BH       ;secteur, sur disque s'il y a lieu
D98E  CD45DA    CALL   0DA45H      ;nombre enreg. Dir à tester dans hl
D991  EB        EX     DE,HL
D992  CDF3DB    CALL   0DBF3H      ;hl=de? testé tous enreg. possibles?
D995  EB        EX     DE,HL
D996  300A      JR     NC,0D9A2H
D998  3E0C      LD     A,0CH
D99A  CD3FDA    CALL   0DA3FH      ;(hl) => buffer valeur de contrôle
D99D  19        ADD    HL,DE
D99E  CDC8D9    CALL   0D9C8H      ;valeur de contrôle sur l'enreg.
D9A1  77        LD     (HL),A      ;et entrer dans buffer valeur contr.
D9A2  C1        POP    BC
D9A3  E1        POP    HL          ;teste si nom de fichier est en
D9A4  CDB8D9    CALL   0D9B8H      ;position correcte
D9A8  D5        PUSH   DE
D9A9  E5        PUSH   HL
D9AA  EB        EX     DE,HL
D9AB  13        INC    DE          ;augmenter nombre entrées occupées
D9AC  3E02      LD     A,02H
D9AE  CD35DA    CALL   0DA35H      ;(hl) => nombre entrées Directory
D9B1  73        LD     (HL),E      ;ranger nouveau nombre
D9B2  23        INC    HL
D9B3  72        LD     (HL),D
D9B4  E1        POP    HL
D9B5  D1        POP    DE
D9B6  37        SCF
D9B7  C9        RET

*************** teste, si position de l'entrée Dir est ok
D9B8  D5        PUSH   DE
D9B9  E5        PUSH   HL          ;nombre entrées Dir
D9BA  3E02      LD     A,02H
D9BC  CD35DA    CALL   0DA35H      ;(hl)=> nombre entrées Dir occupées
D9BF  5E        LD     E,(HL)      ;nombre dans de
D9C0  23        INC    HL
D9C1  56        LD     D,(HL)
D9C2  E1        POP    HL
D9C3  CDF3DB    CALL   0DBF3H      ;hl = de, entrée calculée ok?
D9C6  D1        POP    DE
D9C7  C9        RET
```

```
************* valeur de contrôle sur l'enregistrement, résultat dans accu
D9C8  C5        PUSH    BC
D9C9  E5        PUSH    HL
D9CA  0680      LD      B,80H         ;lonueur enregistrement
D9CC  3E08      LD      A,08H
D9CE  CD3FDA    CALL    0DA3FH        ;(hI) => buffer enregistrement
D9D1  AF        XOR     A             ;vider accu
D9D2  86        ADD     A,(HL)        ;valeur de contrôle sur l'enregistr.
D9D3  23        INC     HL
D9D4  10FC      DJNZ    0D9D2H
D9D6  E1        POP     HL
D9D7  C1        POP     BC
D9D8  C9        RET

********* tester si attribut READ ONLY, bit 7 premier caractère extension
D9D9  E5        PUSH    HL
D9DA  210900    LD      HL,0009H      ;premier caractère extension
D9DD  1804      JR      0D9E3H

*************** tester si attribut SYS, bit 7 second caractère extension
D9DF  E5        PUSH    HL
D9E0  210A00    LD      HL,000AH      ;second caractère extension
D9E3  19        ADD     HL,DE
D9E4  7E        LD      A,(HL)        ;caractère dans accu
D9E5  87        ADD     A,A           ;Carry mis, si attribut mis
D9E6  E1        POP     HL
D9E7  C9        RET

*************** amener enregistrement, No dans de, dans buffer enregistr.
D9E8  C5        PUSH    BC
D9E9  D5        PUSH    DE
D9EA  E5        PUSH    HL
D9EB  CD06DA    CALL    0DA06H        ;déterm.piste et secteur avec No enr.
D9EE  CD4CC5    CALL    0C54CH        ;lire enregistrement dans buffer enr.
D9F1  180B      JR      0D9FEH        ;message d'erreur dans accu

*************** écrire enregistrement, No dans de, dans buffer secteur
D9F3  C5        PUSH    BC
D9F4  D5        PUSH    DE
D9F5  E5        PUSH    HL
D9F6  C5        PUSH    BC            ;déterminer piste et secteur
D9F7  CD06DA    CALL    0DA06H        ;d'après No d'enregistrement
```

```
D9FA  C1        POP     BC
D9FB  CD2EC5    CALL    0C52EH      ;écrire enreg. dans buffer secteur
D9FE  B7        OR      A           ;message d'erreur dans accu
D9FF  C2B6CD    JP      NZ,0CDB6H   ;=> erreur, interrompre instruction
DA02  E1        POP     HL
DA03  D1        POP     DE
DA04  C1        POP     BC
DA05  C9        RET

*************** calcule piste et secteur d'après numéro d'enregistrement
DA06  D5        PUSH    DE          ;numéro d'enregistrement
DA07  44        LD      B,H         ;adresse buffer pour un enregistremnt
DA08  4D        LD      C,L         ;dans bc et transmettre
DA09  CD1AC5    CALL    0C51AH      ;aux routines du controller
DA0C  D1        POP     DE
DA0D  3E0D      LD      A,0DH
DA0F  CD45DA    CALL    0DA45H      ;décalage de piste dans hl
DA12  44        LD      B,H         ;et bc
DA13  4D        LD      C,L
DA14  AF        XOR     A
DA15  CD45DA    CALL    0DA45H      ;Nbre enregistr./piste dans hl
DA18  0B        DEC     BC          ;correction
DA19  03        INC     BC
DA1A  7B        LD      A,E
DA1B  95        SUB     L
DA1C  5F        LD      E,A
DA1D  7A        LD      A,D
DA1E  9C        SBC     A,H
DA1F  57        LD      D,A
DA20  30F7      JR      NC,0DA19H
DA22  19        ADD     HL,DE
DA23  E5        PUSH    HL
DA24  CD24C5    CALL    0C524H      ;piste en c dans be54 pour Controller
DA27  C1        POP     BC
DA28  AF        XOR     A
DA29  CD3FDA    CALL    0DA3FH      ;(hl) => (a910/a920)
DA2C  EB        EX      DE,HL
DA2D  CD5AC5    CALL    0C55AH      ;traduire numéro d'enregistrement
DA30  4D        LD      C,L
DA31  44        LD      B,H
DA32  C329C5    JP      0C529H      ;secteur dans be55 pour Controller
```

```
*************** charge header paramètres disque + accu dans hl
DA35  FD8603     ADD     A,(IY+03H)  ;ajoute octet faible DPH actuel
DA38  6F         LD      L,A         ;résultat dans l
DA39  FD8E04     ADC     A,(IY+04H)  ;ajoute octet fort, éventu.avec Carry
DA3C  95         SUB     L
DA3D  67         LD      H,A         ;octet fort dans h
DA3E  C9         RET

*************** charge contenu (header paramètres disque + accu) dans hl
DA3F  CD35DA     CALL    0DA35H      ;(hl) => DPH + accu
DA42  C3F9DB     JP      0DBF9H      ;ld hl,(hl)

*************** charge contenu (DPB+accu) dans hl
DA45  F5         PUSH    AF
DA46  3E0A       LD      A,0AH
DA48  CD3FDA     CALL    0DA3FH      ;(hl) => début bloc paramètres disque
DA4B  F1         POP     AF          ;décalage voulu
DA4C  85         ADD     A,L
DA4D  6F         LD      L,A
DA4E  8C         ADC     A,H         ;calcule adresse nécessaire
DA4F  95         SUB     L
DA50  67         LD      H,A
DA51  C3F9DB     JP      0DBF9H      ;LD HL,(HL)

*************** charge contenu (bloc paramètres disque + accu) dans accu
DA54  E5         PUSH    HL
DA55  CD45DA     CALL    0DA45H      ;charge contenu (DPB+accu) dans hl
DA58  7D         LD      A,L         ;octet voulu dans accu
DA59  E1         POP     HL
DA5A  C9         RET

***************
DA5B  11E400     LD      DE,00E4H
DA5E  1803       JR      0DA63H

***************
DA60  11F400     LD      DE,00F4H
DA63  0E20       LD      C,20H
DA65  CD74DA     CALL    0DA74H
DA68  182B       JR      0DA95H
```

```
*************** 
DA6A  CD6FDA    CALL    0DA6FH      ;crée EFN, teste nom de fichier
DA6D  1826      JR      0DA95H

*************** crée nom fichier étendu, teste si espace ou '?'
DA6F  0EFF      LD      C,0FFH
DA71  11E400    LD      DE,00E4H
DA74  CDA0DA    CALL    0DAA0H      ;disposer nom de fichier dans buffer
DA77  C5        PUSH    BC
DA78  160B      LD      D,0BH       ;longueur du nom de fichier avec ext.
DA7A  03        INC     BC
DA7B  03        INC     BC          ;bc => nom de fichier
DA7C  0A        LD      A,(BC)      ;premier caractère du nom dans accu
DA7D  FE3F      CP      3FH         ;joker '?'
4A7F  2869      JR      Z,0DAEAH    ;si oui => sortir Bad command
DA81  15        DEC     D
DA82  20F7      JR      NZ,0DA7BH   ;tester si prochain caractère '?'
DA84  C1        POP     BC          ;trouvé aucun point d'interrogation
DA85  C9        RET                 ;semble ok

***************
DA86  0600      LD      B,00H
DA88  CDA6DA    CALL    0DAA6H
DA8B  1808      JR      0DA95H

***************
DA8D  0E20      LD      C,20H       ;espace
DA8F  11E400    LD      DE,00E4H    ;adresse buffer enregistrement
DA92  CDA0DA    CALL    0DAA0H      ;disposer nom de fichier
DA95  210D00    LD      HL,000DH
DA98  09        ADD     HL,BC
DA99  36FF      LD      (HL),0FFH
DA9B  23        INC     HL
DA9C  23        INC     HL
DA9D  36FF      LD      (HL),0FFH
DA9F  C9        RET

*************** crée nom fichier étendu, teste si nom de fichier ok
DAA0  CDB6DA    CALL    0DAB6H      ;crée nom de fichier étendu
DAA3  2845      JR      Z,0DAEAH    ;=>espace ds nom fichier, Bad Command
DAA5  C9        RET
```

```
***************
DAA6  0E20     LD     C,20H        ;espace
DAA8  11E400   LD     DE,00E4H     ;décalage par rapport buffer enreg.
DAAB  CDB6DA   CALL   0DAB6H
DAAE  C5       PUSH   BC
DAAF  0E0B     LD     C,0BH
DAB1  CC8EDB   CALL   Z,0DB8EH
DAB4  C1       POP    BC
DAB5  C9       RET

***************
DAB6  E5       PUSH   HL
DAB7  CD98CA   CALL   0CA98H       ;de=de+iy, buffer nom fichier étendu
DABA  D5       PUSH   DE           ;ranger
DABB  FD7E00   LD     A,(IY+00H)   ;numéro du lecteur actif
DABE  12       LD     (DE),A       ;entrer dans nom fichier
DABF  13       INC    DE
DAC0  FD7E01   LD     A,(IY+01H)   ;numéro user actif
DAC3  12       LD     (DE),A       ;entrer dans nom de fichier
DAC4  13       INC    DE
DAC5  C5       PUSH   BC
DAC6  41       LD     B,C
DAC7  0E08     LD     C,08H        ;nom fichier a 8 caractères de long
DAC9  CD85DB   CALL   0DB85H       ;8 espaces dans (de) à (de+7)
DACC  78       LD     A,B
DACD  0E03     LD     C,03H        ;extension a 3 caractères de long
DACF  CD90DB   CALL   0DB90H       ;3 fois ffh dans (de+8h) à (de+0ah)
DAD2  010300   LD     BC,0003H     ;
DAD5  CDAFCA   CALL   0CAAFH       ;vide (de+0bh) à (de+0dh)
DAD8  C1       POP    BC           ;b=> taille nom fichier ent, c=> 0ffh
DAD9  D1       POP    DE           ;de => début nom fichier étendu (EFN)
DADA  E1       POP    HL           ;hl => adr. nom fichier entrée (IFN)
DADB  D5       PUSH   DE
DADC  CDEDDA   CALL   0DAEDH       ;crée EFN, si IFN correct
DADF  D1       POP    DE
DAE0  3008     JR     NC,0DAEAH    ;bad command,nom fich. entrée pas ok
DAE2  42       LD     B,D          ;adresse EFN dans bc
DAE3  4B       LD     C,E
DAE4  13       INC    DE           ;de => nom de fichier
DAE5  13       INC    DE
DAE6  1A       LD     A,(DE)       ;premier caractère du EFN
DAE7  FE20     CP     20H          ;espace?
```

```
DAE9  C9        RET

***************
DAEA  C3AFCD    JP      0CDAFH       ;sortir Bad command, interr.instr.

*************** teste IFN, crée EFN avec Drive et USER
DAED  2B        DEC     HL           ;Dummy
DAEE  CD97DB    CALL    0DB97H       ;tester longueur IFN
DAF1  3F        CCF
DAF2  D8        RET     C            ;erreur apparue! pas de nom fichier?
DAF3  4F        LD      C,A
DAF4  E5        PUSH    HL
DAF5  C5        PUSH    BC
DAF6  FE3A      CP      3AH          ;caractère est un ':'?
DAF8  2806      JR      Z,0DB00H     ;oui, alors =>
DAFA  CDA5DB    CALL    0DBA5H       ;un caract. de IFN, convert.majusc.
DAFD  38F7      JR      C,0DAF6H     ;tester si ':'
DAFF  37        SCF                  ;trouvé aucun ':' dans le nom

*************** si ':', tester si numéro USER valable
DB00  C1        POP     BC
DB01  E1        POP     HL
DB02  79        LD      A,C
DB03  383E      JR      C,0DB43H     ;saut si trouvé aucun ':'
DB05  13        INC     DE           ;
DB06  FE30      CP      30H          ;'0'
DB08  381F      JR      C,0DB29H     ;caractère < '0', donc pas chiffre
DB0A  FE3A      CP      3AH          ;':'
DB0C  301B      JR      NC,0DB29H    ;caractère >= ':', pas chiffre
DB0E  D630      SUB     30H          ;est un chiffre,-30h donne nb binaire
DB10  4F        LD      C,A          ;
DB11  12        LD      (DE),A       ;entrer chiffre dans EFN
DB12  CDA5DB    CALL    0DBA5H       ;un caract. de IFN, convert.majusc.
DB15  FE30      CP      30H          ;'0'
DB17  3810      JR      C,0DB29H     ;caractère < '0', alors pas chiffre
DB19  FE3A      CP      3AH          ;':'
DB1B  300C      JR      NC,0DB29H    ;caractère >= ':', pas chiffre
DB1D  B7        OR      A            ;caractère est un chiffre
DB1E  0D        DEC     C            ;premier chiffre est un 1?
DB1F  C0        RET     NZ           ;non, erreur en IFN, faux No USER
DB20  C6DA      ADD     A,0DAH       ;tester si second chiffre de 0 à 6
DB22  FE10      CP      10H
```

```
DB24  D0        RET    NC          ;second chiffre >6, faux No USER
DB25  12        LD     (DE),A      ;entrer chiffre dans EFN
DB26  CDA5DB    CALL   0DBA5H      ;un caractère de IFN, convert.majus.

************* si pas chiffre, tester (logiquement) si No lecteur valable
DB29  1B        DEC    DE
DB2A  FE51      CP     51H         ;'Q'
DB2C  300A      JR     NC,0DB38H   ;Caract. est 'Q','R'... erreur ds IFN
DB2E  FE41      CP     41H         ;'A'
DB30  3806      JR     C,0DB38H    ;Caractère < 'A', erreur dans IFN
DB32  D641      SUB    41H         ;ôter 'A', donne 0 à 15
DB34  12        LD     (DE),A      ;ce sont les lecteurs possibles
                                    (logiquement)
DB35  CDA5DB    CALL   0DBA5H      ;un caractère de IFN, convert.majusc.

*************** si USER et/ou Drive, alors ':' doit suivre, sinon erreur
DB38  CD9BDB    CALL   0DB9BH      ;retirer caractère, ignorer espace
DB3B  EE3A      XOR    3AH         ;maintenant doit venir un ':'
DB3D  C0        RET    NZ          ;si non, alors erreur dans IFN
DB3E  CD97DB    CALL   0DB97H      ;retirer caractère, ignorer espace
DB41  3F        CCF
DB42  D8        RET    C           ;erreur, pas de caractère après ':'

*************** détermine nom fichier effectif et extension d'après IFN
DB43  13        INC    DE
DB44  13        INC    DE
DB45  FE2E      CP     2EH         ;'.'
DB47  C8        RET    Z           ;nom de fichier manque, entré qu'ext.
DB48  0E08      LD     C,08H       ;longueur nom de fichier étendu (EFN)
DB4A  CD58DB    CALL   0DB58H
DB4D  D8        RET    C
DB4E  EE2E      XOR    2EH         ;maintenant doit venir un '.'
DB50  C0        RET    NZ          ;erreur, IFN trop long
DB51  CD97DB    CALL   0DB97H      ;retirer caractère, ignorer espace
DB54  0E03      LD     C,03H       ;longueur de l'extension
DB56  302D      JR     NC,0DB85H   ;plus de caract.,alors ext.=3 espaces

*************** teste caractère pour nom fichier et ext., entre dans EFN
DB58  FE20      CP     20H         ;' '
DB5A  3829      JR     C,0DB85H    ;si pas d'espace,remplir avec espaces
DB5C  E5        PUSH   HL
DB5D  C5        PUSH   BC
```

```
DB5E  47        LD     B,A         ;sauvegarde provisoire
DB5F  21B2DB    LD     HL,0DBB2H   ;table des caractères 'interdits'
DB62  7E        LD     A,(HL)      ;retirer valeur de table
DB63  23        INC    HL          ;augmenter pointeur
DB64  B7        OR     A
DB65  2804      JR     Z,0DB6BH    ;table terminée?
DB67  B8        CP     B           ;comparer valeur table avec caract.
DB68  20F8      JR     NZ,0DB62H   ;si différent, comparer suivante
DB6A  37        SCF                ;marque caractère 'interdit'
DB6B  78        LD     A,B         ;caractère dans accu
DB6C  C1        POP    BC
DB6D  E1        POP    HL
DB6E  3815      JR     C,0DB85H    ;caract.'interdit',reste avec espaces
DB70  0D        DEC    C           ;compteur longueur nom ou extension
DB71  F8        RET    M           ;tous caractères parcourus? alors =>
DB72  FE2A      CP     2AH         ;'*', joker pour le reste
DB74  CC8EDB    CALL   Z,0DB8EH    ;si joker alors remplir reste de '?'
DB77  12        LD     (DE),A      ;caractère ok. Entrer dns nom fichier
DB79  CDA5DB    CALL   0DBA5H      ;caract.de nom fichier,convert.majus.
DB7C  3007      JR     NC,0DB85H   ;plus de caractère, remplir reste
                                    avec espaces
DB7E  FE20      CP     20H         ;caractère espace?
DB80  20D6      JR     NZ,0DB58H   ;si non, alors tester et entrer

*************** jeter rester dans IFN après espace
DB82  CD9BDB    CALL   0DB9BH      ;teste si espace, si oui, car.suivant

*************** remplit mémoire (de) à (de+c) avec caractère espace
DB85  F5        PUSH   AF
DB86  3E20      LD     A,20H       ;valeur pour caractère espace
DB88  CD90DB    CALL   0DB90H
DB8B  F1        POP    AF
DB8C  3F        CCF
DB8D  C9        RET

*************** remplit mémoire (de) à (de+c) avec '?'
DB8E  3E3F      LD     A,3FH       ;'?', joker pour un caractère

*************** remplit mémoire (de) à (de+c) avec caractère dans accu
DB90  0C        INC    C           ;correction nécessaire
DB91  0D        DEC    C           ;case mémoire suivante
DB92  C8        RET    Z           ;ou éventuellement c=0, alors fin
```

```
DB93  12          LD      (DE),A       ;caractère dans mémoire adressée
DB94  13          INC     DE           ;augmenter pointeur
DB95  18FA        JR      0DB91H       ;et encore une fois

*************** retire un caractère, teste si longueur nom de fichier > 0
DB97  CDA5DB      CALL    0DBA5H       ;un caractère, teste longueur nom
DB9A  D0          RET     NC           ;longueur 0, erreur!

*** teste si accu est espace, si espace, alors retirer prochain caractère
DB9B  FE20        CP      20H          ;espace?
DB9D  37          SCF
DB9E  C0          RET     NZ           ;c'était quelque chose d'autre
DB9F  CDA5DB      CALL    0DBA5H       ;retirer prochain caractère
DBA2  38F7        JR      C,0DB9BH     ;teste si espace, si oui,
DBA4  C9          RET                  ;prochain caractère

******** retire un caractère du nom de fichier et convertit en majuscules
DBA5  78          LD      A,B          ;longueur nom fichier (restante)
DBA6  B7          OR      A            ;longueur <> 0?
DBA7  C8          RET     Z            ;si 0, alors Return
DBA8  23          INC     HL           ;(hl) => pointeur sur nom de fichier
DBA9  05          DEC     B            ;d'entrée
DBAA  E7          RST     20H          ;RAM LAM, LD A,(HL) de Ram
DBAB  E67F        AND     7FH          ;que caractère ASCII, S.V.P
DBAD  CDA6CA      CALL    0CAA6H       ;convertir en majuscules
DBB0  37          SCF                  ;marque octet retiré
DBB1  C9          RET

*************** caractères interdits dans nom de fichier
DBB2  3C          DEFB    3CH          ;'<'
DBB3  3E          DEFB    3EH          ;'>'
DBB4  2E          DEFB    2EH          ;'.'
DBB5  2C          DEFB    2CH          ;','
DBB6  3B          DEFB    3BH          ;';'
DBB7  3A          DEFB    3AH          ;':'
DBB8  3D          DEFB    3DH          ;'='
DBB8  5B          DEFB    5BH          ;crochet ouvert
DBB8  5D          DEFB    5DH          ;crochet fermé
DBB8  5F          DEFB    5FH          ;soulignage
DBB8  25          DEFB    25H          ;'%'
DBB8  7C          DEFB    7CH          ;Shift arobas
```

```
DBBE  28        DEFB    28H         ;'('
DBBE  29        DEFB    29H         ;')'
DBBE  2F        DEFB    2FH         ;'/'
DBBE  5C        DEFB    5CH         ;Backslash
DBBE  7F        DEFB    7FH         ;Delete
DBBE  00        DEFB    00H         ;marque fin de la table

*************** sort trois espaces, No lecteur actuel dans c
DBC4  3E01      LD      A,01H       ;message système 1
DBC6  1802      JR      0DBCAH

*************** sort 'nom de fichier', No lecteur actuel dans c
DBC8  3E0B      LD      A,0BH       ;message système 11
DBCA  C5        PUSH    BC
DBCB  FD4E02    LD      C,(IY+02H)  ;No lecteur dans c
DBCE  180A      JR      0DBDAH

*************** sort 'Drive #: user #', e=lecteur, c= No user
DBD0  C5        PUSH    BC
DBD1  0A        LD      A,(BC)      ;numéro lecteur
DBD2  5F        LD      E,A         ;dans e
DBD3  1600      LD      D,00H
DBD5  0B        DEC     BC
DBD6  0A        LD      A,(BC)      ;No user
DBD7  4F        LD      C,A         ;dans c
DBD8  3E0C      LD      A,0CH       ;message système 12
DBDA  CDEBCA    CALL    0CAEBH      ;sortir message système
DBDD  C1        POP     BC
DBDE  C9        RET

*************** transfère 32 octets avec LDIR de hl dans de
DBDF  E5        PUSH    HL
DBE0  D5        PUSH    DE
DBE1  C5        PUSH    BC
DBE2  012000    LD      BC,0020H
DBE5  EDB0      LDIR
DBE7  C1        POP     BC
DBE8  D1        POP     DE
DBE9  E1        POP     HL
DBEA  C9        RET
```

```
*************** divise hl par (accu^2)
DBEB  CB3C      SRL     H
DBED  CB1D      RR      L
DBEF  3D        DEC     A
DBF0  20F9      JR      NZ,0DBEBH
DBF2  C9        RET

*************** compare hl avec de
DBF3  E5        PUSH    HL
DBF4  B7        OR      A
DBF5  ED52      SBC     HL,DE
DBF7  E1        POP     HL
DBF8  C9        RET

*************** charge (hl) dans hl
DBF9  D5        PUSH    DE
DBFA  5E        LD      E,(HL)
DBFB  23        INC     HL
DBFC  56        LD      D,(HL)
DBFD  EB        EX      DE,HL
DBFE  D1        POP     DE
DBFF  C9        RET

*************** dc00 à dfff ne sont pas utilisés
DC00  à   DFFF  RST     38H
```

5.1 ERREUR AVEC MERGE ET CHAIN MERGE

Ce chapitre intéressera tous ceux qui ont été parmi les premiers à acheter un lecteur de disquette. Une erreur s'est malheureusement glissée dans le système d'exploitation de sorte que lorsqu'on charge un programme supplémentaire avec les instructions MERGE et CHAIN MERGE l'erreur 'EOF met' survient de façon injustifiée.
Comme le fabricant a connaissance de cette erreur, celle-ci sera éliminée dans les versions ultérieures du système d'exploitation. Ceux d'entre vous qui ne savent pas encore si leur lecteur de disquette comporte ou non cette erreur pourront le savoir en étudiant le présent chapitre.

Comme nous l'avons déjà indiqué et comme vous l'avez peut-être déjà expérimenté par vous-même, le système d'exploitation AMSDOS comporte une erreur. Ne soyez pas trop irrité par ce fait car même les installations les plus coûteuses comportent encore d'innombrables erreurs dans leur système d'exploitation.

Il n'y a pas non plus là de raison de paniquer: votre problème est déjà résolu puisque vous avez le présent ouvrage entre les mains. Nous ne vous fournissons pas UNE possibilité de solution, mais DEUX. Mais quand et pourquoi cette erreur se produit-elle? Peut-être n'avez vous eu jusqu'ici aucun problème lors de l'utilisation des instructions MERGE et CHAIN MERGE. C'est tout à fait possible. Notamment pour les petites routines, il est même assez improbable qu'une erreur se produise.
Essayez donc notre exemple:

```
NEW

10 PRINT x;"C'est réussi."
20 GOTO 20


SAVE "Deux"

NEW
```

```
10 PRINT "Nous essayons ..."
20 x=1
30 CHAIN MERGE "Deux"
```

RUN

Vous pouvez maintenant vous faire LISTer le programme. Tout s'est passé comme voulu et attendu. (Voyez également le chapitre correspondant du manuel du Basic CPC).
Entrez maintenant:

NEW

```
26 PRINT x;"C'est réussi."
27 END
```

SAVE "Deux"

NEW

```
10 PRINT "Nous essayons ..."
20 x=1
25 CHAIN MERGE "Deux"
```

RUN

Faites toutefois très attention à bien respecter les numéros de ligne que nous vous indiquons car ils sont importants pour notre exemple.

Après avoir lancé le programme, vous obtiendrez un

EOF met in 25

Vous savez que EOF signifie End of File et vous vous demandez certainement pourquoi l'ordinateur a rencontré une fin de fichier? C'est exactement en cela qu'est l'erreur du système d'exploitation. Nous avons provoqué cette erreur dans notre exemple en utilisant le numéro de ligne 26. Dès que, lors du chargement d'un second programme, un &1a = 26 est rencontré, le message d'erreur 'EOF met' est sorti. Le code 26 ne peut pas se présenter comme caractère ASCII dans un programme, mais il peut

par contre être utilisé dans le codage des programmes.

Lors de la sauvegarde d'un programme sur disquette est écrite également sur disquette pour chaque ligne du programme la longueur ainsi que l'adresse de début en mémoire de cette ligne. Cela représente trois caractères pour chaque ligne et le risque est grand qu'un de ces trois octets vaille &1a. D'autre part le numéro de ligne est sauvé sous la forme d'un nombre entier sur 16 bits. Ce nombre peut également comporter un &1a, comme dans notre exemple. La probabilité est donc de 1:(256/5)=51.2 par ligne, ce qui signifie que statistiquement, un 'EOF met' doit survenir au plus tard lorsque le programme dépasse 51 lignes. Dans la pratique cependant, un nombre de lignes très inférieur suffit souvent.

Une première solution possible consiste à sauvegarder les programmes à charger en second sous la forme de ce qu'on appelle les fichiers ASCII. Les programmes ne sont plus alors codés mais sont écrits sur la disquette comme à l'écran, caractère par caractère. Les fichiers de programmes deviennent ainsi un peu plus longs, un programme d'une taille de 30 Koctets occupera par exemple 32 Koctets sur la disquette. Dans notre exemple, vous auriez donc dû entrer:

```
SAVE "Deux",a
```

Le ',a' est mis pour fichier ASCII. Cette première solution est aussi la plus simple. Si vous avez toutefois des programmes protégés que vous voulez charger en second (par exemple), cette solution n'est pas utilisable.

Nous avons donc un petit programme à vous offrir comme deuxième solution. Ce programme exécute l'instruction CHAIN MERGE comme elle devrait normalement être exécutée, c'est-à-dire comme elle est décrite dans le manuel d'utilisation et comme elle fonctionne en Basic cassette.

```
1 '
2 ' Firmware Patch for CHAIN-MERGE
3 ' Amstrad CPC & DDI-1
4 '
5 '
6 '
```

```
10 MEMORY HIMEM-41
20 DEF FNmsb(a)=&FF AND INT(a/256)
30 DEF FNlsb(a)=&FF AND UNT(a)
40 FOR i=HIMEM+1 TO HIMEM+38
50 READ byte
60 POKE i,byte
70 NEXT i
80 POKE HIMEM+3, FNlsb(HIMEM+39)
90 POKE HIMEM+4, FNmsb(HIMEM+39)
100 POKE HIMEM+9, FNIsb(HIMEM+41)
110 POKE HIMEM+10,FNmsb(HIMEM+41)
120 POKE HIMEM+18,FNlsb(HIMEM+1)
130 POKE HIMEM+19,FNmsb(HIMEM+1)
140 'CAS IN CHAR
150 POKE HIMEM+39, PEEK(&BC80+0)
160 POKE HIMEM+40, PEEK(&BC80+1)
170 POKE HIMEM+41, PEEK(&BC80+2)
180 POKE &BC80+0, &C3
190 POKE &BC80+1, FNlsb(HIMEM+1)
200 POKE &BC80+2, FNmsb(HIMEM+1)
210 DATA &e5,&2A,&00,&00,&22,&80,&bc
220 DATA &3a,&00,&00,&32,&82,&bc
230 DATA &cd,&80,&bc,&21,&00,&00
240 DATA &22,&81,&bc,&21,&80,&bc
250 DATA &36,&c3,&e1,&d8,&c8,&fe,&1a
260 DATA &37,&3f,&c0,&b7,&37,&c9
```

Une fois cette routine entrée, sauvez-la sur disquette. Lorsque vous aurez ensuite un programme qui utilise l'instruction CHAIN MERGE, vous pourrez placer cette routine au début du programme. Une fois la routine exécutée, vous pouvez à nouveau effacer les lignes correspondant à cette routine avec DELETE (cela est également possible dans un programme).

5.2 MESSAGES D'ERREUR

Il y a en certainement parmi vous que le simple terme de 'message d'erreur' suffit à plonger dans une colère noire parce qu'ils ne peuvent manquer de se rappeler telle ou telle vaine tentative pour créer un programme qui soit à l'abri des messages d'erreur du lecteur de disquette.

Les messages d'erreur sont cependant une chose tout à fait naturelle lorsqu'on travaille avec des ordinateurs. Tout le monde fait des erreurs que l'ordinateur affiche alors à l'écran. Si l'on veut par exemple charger un fichier qui n'existe pas sur la disquette, on obtient la remarque lapidaire suivante:

Filename.Ext not found

Vous savez alors que vous avez mal indiqué le nom du fichier ou que vous n'avez pas placé dans le lecteur la bonne disquette. Le lecteur de disquette se donne dans un tel cas beaucoup de mal pour essayer de vous servir, avant de sortir le message d'erreur. Si aucune extension (type de fichier) n'est indiquée, le lecteur de disquette DDI-1 effectuera TROIS tentatives pour essayer de trouver le fichier voulu.

1) sous le nom de Filename.
2) sous le nom de Filename.BAS
3) sous le nom de Filename.BIN

Ce n'est que si toutes ces tentatives échouent que le lecteur de disquette vous enverra le message d'erreur indiqué plus haut. Si toutefois vous indiquez l'extension avec le nom de fichier, une seule tentative sera effectuée. (Par exemple LOAD"Jeuxdeba.lle" ou OPENIN"Donnees.dat" etc.)

Nous vous expliquons par ailleurs au chapitre 1.5 comment éviter certaines erreurs. Ces astuces de programmation ne peuvent cependant pas toujours être mises en oeuvre avec succès.

Il y a malheureusement des instructions telles que Disc missing ou d'autres semblables que l'on ne peut absolument pas intercepter par programme.

Cette erreur ne signifie pas nécessairement que le lecteur de disquette soit vide, il se peut également que la disquette n'ait pas été complètement enfoncée. Vous obtenez alors le message d'erreur:

Drive A: disc missing

Retry, Ignore or Cancel?

Un autre exemple: si votre disquette n'est plus en très bon état et qu'un Read Fail apparaît comme message d'erreur. On pourrait encore allonger cette liste de possibilités. Tout cela ne serait d'ailleurs pas si dramatique et ces messages d'erreur pourraient même n'être considérés que comme un grand avantage de ce lecteur de disquette, s'il n'y avait pas un inconvénient décisif! Si en effet le lecteur de disquette détecte une erreur lors du déroulement d'un programme, quelle que soit l'erreur, le texte de cette erreur vous est affiché sur l'écran et le déroulement du programme est immédiatement interrompu. Ce n'est pas grave, direz-vous, cela m'arrive bien aussi lorsqu'il y a une erreur dans mon programme. Il est vrai, cependant:

Il est possible d'éliminer presque totalement les erreurs d'un programme. On peut par exemple éliminer toutes les erreurs de syntaxe, tester la validité des entrées faites par l'utilisateur et intercepter les mauvaises entrées. Il est ainsi possible d'intercepter pratiquement toutes les erreurs imaginables lors du déroulement d'un programme.
Il est en outre possible d'utiliser l'instruction Basic ON ERROR GOTO afin qu'un message d'erreur survenant pendant le déroulement du programme n'ait pas pour effet de l'interrompre mais seulement de SUSPENDRE son exécution.
Lorsqu'une erreur se produit, le déroulement normal du programme est alors suspendu, l'endroit où l'erreur s'est produite est mémorisé et l'ordinateur saute à une sous-routine qui pourra alors éventuellement réagir à cette erreur. En programmant de la sorte, on peut intercepter à 100% tout risque d'interruption du programme.

Mais que pourriez-vous faire pour que le programme puisse s'assurer qu'une disquette figure bien dans le lecteur? Vous pouvez indiquer à l'utilisateur du programme de placer une disquette dans le lecteur mais cela n'exclut pas un tas d'autres messages d'erreur possibles. Bien, direz-vous, je peux donc utiliser l'instruction ON ERROR GOTO qui est vraiment pratique.

Malheureusement ON ERROR GOTO n'a aucun effet pour les messages d'erreur du lecteur de disquette. Les messages d'erreur sont sortis malgré tout et le programme est interrompu.

(A ceux d'entre vous qui ne connaissent pas encore suffisamment cette instruction, nous recommandons d'en étudier la description dans le manuel d'utilisation.)

Il convient de noter cependant ici que le CPC 664 vous offre la possibilité d'intercepter les messages d'erreur du disque avec l'instruction ON ERROR GOTO. La variable d'erreur système ERR a alors la valeur 32 et la variable DERR contient en outre le numéro d'erreur transmis par le DOS. Le programme n'est donc pas interrompu mais le texte d'erreur apparaît sur l'écran. Un autre inconvénient est qu'on ne peut demander clairement de QUELLE erreur il s'agit car ERR contient toujours 32 et la variable DERR ne donne pas beaucoup de renseignements à ce sujet.

Les bons conseils sont donc très appréciables dans ce domaine. Même le plus opiniâtre des programmeurs peut en effet être un jour contraint à l'abandon. Imaginez comme il est désespérant en effet d'entrer 200 enregistrements et au moment de les sauvegarder de recevoir de la disquette le message

Disc full

Le programme est alors interrompu et vous pouvez entrer à nouveau vos 200 enregistrements! Un tel programme ne correspond bien sûr pas vraiment à l'image qu'on se fait d'un logiciel professionnel. Protéger les programmes contre un erreur d'utilisation est en effet le premier souci des programmeurs professionnels. Mais le lecteur de disquette est justement très sujet à erreur.

Vous trouverez donc dans ce chapitre une routine en langage machine qui résout définitivement ce problème. Les erreurs ne sont plus sorties sur l'écran et les programmes ne sont plus interrompus. On aurait pu faire en sorte que le programmeur ne remarque même pas qu'une erreur a été annoncée par le lecteur de disquette. Mais cela n'est vraiment pas souhaitable non plus car il vaut mieux prendre connaissance de l'erreur, l'important étant simplement que le programme ne soit pas alors interrompu. Avec cette routine le problème a été résolu de façon à ce

qu'une erreur du lecteur de disquette entraîne la sortie du message d'erreur

Unknown user function

Ce message d'erreur survient en fait très rarement et c'est pourquoi il a été choisi de façon à ce que les erreurs disque puissent être distinguées dans un programme des autres erreurs. Tous les messages d'erreur ont des codes d'erreur qu'on peut interroger avec la variable système ERR. On peut également interroger la ligne dans laquelle est survenue la dernière erreur. On utilise à cet effet la variable système ERL.

Quand donc un message d'erreur est envoyé par le lecteur de disquette, celui-ci est intercepté et stocké provisoirement dans un buffer spécial. Le programme n'est pas interrompu et une erreur UNKNOWN USER FUNCTION avec le code d'erreur 18 est générée. Cette erreur peut alors fort bien être interceptée avec l'instruction ON ERROR GOTO. Il faut ensuite examiner dans la routine d'erreur correspondante quelle erreur a été annoncée. Si la variable de code d'erreur ERR vaut 18, c'est qu'on est en présence d'une erreur disque. On peut alors se faire transmettre le texte par la routine, dans une variable alphanumérique, pour déterminer de quelle erreur précise il s'agit. On peut alors réagir dans le programme à l'erreur annoncée. Il est possible d'activer ou de désactiver cette routine de protection à tout moment.

Un programme d'exemple

```
10 ON ERROR GOTO 1000
20 CALL active : 'Routine d'erreur activée
30 OPENIN "nombres"
40 WHILE NOT EOF
50   INPUT #9,a
60   PRINT a,
70 WEND
80 CLOSEIN
90 CALL desactive : END
1000 IF ERR<>18 THEN RESUME NEXT
1010 ds$="+" : CALL msg,@ds$
1020 PRINT "Disk:";ds$
1030 RESUME 90
```

Vous voyez que les routines d'erreur sont appelées avec l'instruction CALL. Il y a là trois différentes adresses d'entrée:

CALL active — Avec Call active, vous activez la routine d'interception des erreurs. Un grand nombre de vecteurs du système d'exploitation sont alors DETOURNES. Ce problème ne peut pas être résolu autrement. Immédiatement après, les messages d'erreur ne sont plus sortis sur l'écran mais interceptés et entrestockés. Après un message d'erreur annoncé par le lecteur de disquette, une erreur Unknown user function est générée. Cette erreur peut être interceptée avec ON ERROR GOTO.

CALL msg,@a$ — Le dernier message d'erreur annoncé est transmis à travers la variable alphanumérique @a$ (toute autre variable alphanumérique est bien sûr également possible). Faites attention à ce que, avant d'être appelée avec a$, la variable alphanumérique a$ ait bien été INITIALISEE. Un a$="" est suffisant. Sinon vous obtiendrez un Improper argument.

CALL desactive — La routine d'interception des erreurs est à nouveau désactivée. L'état normal est à nouveau rétabli. Immédiatement après les messages d'erreur apparaissent à nouveau sur l'écran. Il est recommandé de placer la désactivation des routines d'erreur à la fin de tout programme car vous risqueriez sinon de vous demander pourquoi un FILE NOT FOUND ne vous est plus communiqué comme d'habitude.

On suppose bien entendu que la routine d'erreur soit en mémoire!

Explication du programme d'erreur:

LIGNE	EXPLICATION
10	On décide qu'après un message d'erreur le programme doit se poursuivre en ligne 1000.
20	La routine d'erreur est activée. Immédiatement après les

messages d'erreur du lecteur de disquette ne sont plus affichés à l'écran mais interceptés et stockés dans des buffers.

30 Le fichier séquentiel "nombres" est ouvert. Si ce fichier n'existe pas, un "File not found" sera généré.

40-80 Si le fichier a pu être trouvé, les nombres sont lus et sortis.

90 La routine d'interception des erreurs est à nouveau désactivée et le programme est terminé.

1000 On teste ici s'il s'agit d'une erreur disquette. Si ce n'est pas le cas, l'exécution du programme se poursuit immédiatement après l'erreur.

1010 La variable ds$ est initialisée. Cela est nécessaire pour ne pas obtenir un "Improper argument" avec @ds$. Le texte d'erreur fourni est alors tranmis à la variable ds$.

1020 Le texte transmis est sorti sur l'écran. Cela ne représente qu'une des possibilités d'utilisation parmi des milliers.

1030 Le programme se poursuit en ligne 90.

Voilà donc un exemple d'application. Vous pouvez également voir d'après le programme de gestion de fichier que nous vous donnons dans ce chapitre comment on peut travailler avec la routine d'erreur de façon très intéressante.

Il s'est avéré très pratique et très intéressant de fixer et d'interroger dans le programme ce qu'on appelle des flags. On peut alors reconnaître à l'intérieur du programme si une erreur s'est produite ou non et réagir comme il convient.

Cela pourrait se présenter comme dans l'exemple suivant:

```
10 ON ERROR GOTO 1000
20 CALL active : 'Routine d'erreur activée
30 errflg=0 : OPENIN "nombres" : IF errflg THEN 100
40 WHILE NOT EOF
50   INPUT #9,a
60   PRINT a,
```

```
70 WEND
80 CLOSEIN
90 CALL desactive : END
100 IF RIGHT$(ds$,9)="not found" THEN REM Réagir au message
110 PRINT "Disk -- ";ds$
120 END
1000 IF ERR<>18 THEN RESUME NEXT
1010 ds$="+" : CALL msg,@ds$
1020 errflg=1
1030 RESUME NEXT
```

Vous voyez dans cet exemple qu'après l'instruction OPENIN on teste s'il y a une erreur. S'il n'y a pas d'erreur, l'exécution normale du programme se poursuit.

Si toutefois il y a une erreur, on teste alors s'il s'agit d'une erreur "File not found". Si c'est un "File not found", on peut y réagir de manière appropriée (par exemple en changeant le nom de fichier). Sinon le message d'erreur sera sorti. On pourrait bien sûr faire ici encore d'autres distinctions. On pourrait ainsi par exemple lors d'un Drive A: disc missing demander à l'utilisateur par un message en clair de placer une disquette dans le lecteur.

Ce mode de traitement des erreurs permet par exemple de déterminer par un essai d'ouverture d'un fichier si le fichier existe déjà ou non et donc de le créer si nécessaire.

Vous voyez que les possibilités sont très diverses et leur mise en oeuvre extrêmement simple.

Vous pouvez placer la routine d'interception des erreurs dans n'importe quelle zone de la mémoire. Pour ceux d'entre vous qui programment en langage machine, nous vous fournissons un listing assembleur. Pour tous les autres nous vous présentons un chargeur Basic pour placer la routine machine dans n'importe quelle zone de la mémoire. La routine a 229 octets de long (ce qui est très peu). C'est certainement une infime dépense de place mémoire par rapport à l'usage qu'on peut en faire. Vous pouvez entrer en ligne 70 du chargeur Basic l'adresse de départ que doit avoir le programme. Si vous avez un programme dans lequel aucune instruction Symbol after n'est utilisée, vous pouvez placer la routine tout en haut de la mémoire, pour perdre le moins de place possible pour votre programme Basic.

En Basic cassette, vous disposez de 43533 octets pour vos programmes et vos données. La limite supérieure de la mémoire est en &AB7F. Lorsque vous connectez le lecteur de disquette, un peu de mémoire est perdue pour les buffers d'entrée et de sortie ainsi que pour le DOS. Après la mise sous tension, HIMEM est en &A67B et vous n'avez plus que 42249 octets de libres. Le DOS nécessite toutefois encore 4096 octets pour les buffers d'entrée et sortie que nous venons d'indiquer. Ces buffers peuvent cependant être librement déplacés (contrairement à la mémoire système). Nous mettons en place cette mémoire avec:

```
OPENOUT "dummy"
MEMORY HIMEM-1
CLOSEOUT
```

Nous obtenons maintenant pour HIMEM la valeur de &967A. Il reste 38152 octets libres. La limite supérieure pour Basic se trouve donc dans le meilleur des cas en &967A. Notre routine a 229 octets de long et nécessite en outre encore 40 octets comme buffer pour le texte du message d'erreur. Nous calculons donc &967A-229-40=956D et nous faisons maintenant commencer notre routine en &956D. Entrez cette valeur dans la ligne 70 du chargeur.
Si vous utilisez toutefois l'instruction Symbol after, vous devrez placer la routine un peu plus bas dans la mémoire. A cet effet, éteignez votre ordinateur et rallumez-le. Entrez alors:

```
Symbol After n
```

Remplacez ici n par la plus grande valeur survenant dans le programme. Entrez ensuite:

```
OPENOUT "dummy"
MEMORY HIMEM-1
CLOSEOUT

PRINT HEX$(HIMEM-229-40)
```

Vous obtiendrez par exemple avec Symbol after 190 un résultat de &93DD. Vous n'avez plus alors qu'à définir &93D0 comme adresse de départ. C'est tout.

Créez de cette manière, à titre d'exercice, une routine située en &9650. Vous pourrez utiliser cette routine dans la plupart des programmes, tant que vous n'utilisez pas d'instructions Symbol after et tant que d'autres

routines machine ne sont pas intégrées dans le programme. Sauvegardez alors ce fichier sur disquette car il est plus facile de charger la routine d'interception des erreurs à partir de la disquette plutôt que d'intégrer chaque fois toutes les lignes DATA dans le programme. Si vous voulez maintenant intégrer la routine d'interception des erreurs dans un programme, conformez-vous à la marche à suivre ci-dessous:

1) Créez le fichier de programme binaire avec le programme de chargeur Basic et sauvegardez ce programme sur disquette (bien sûr sur la même disquette sur laquelle est sauvegardé le programme qui nécessite et charge cette routine).

2) Intégrez la routine dans le programme. Il convient ici aussi de procéder dans l'ordre car il peut sinon arriver que 4096 octets soient inutilement perdus. Cela arrive en effet lorsque la routine est d'abord chargée, que la limite supérieure de la mémoire est alors abaissée et que le buffer n'est défini qu'ensuite.

Un programme chargeant la routine d'interception des erreurs devrait se présenter à peu près comme ceci:

```
10 ON ERROR GOTO 20000 'Routine d'erreur
20 LOAD "ERROR.BIN"
30 MEMORY &9650-1
40 OPENOUT "dummy"
50 MEMORY HIMEM-1
60 CLOSEOUT
70 .
80 .
90 .
```

Vous reconnaissez l'ordre à suivre. La routine doit d'abord être générée et sauvée sur disquette sous le nom "Error.bin"; n'importe quel autre nom est également possible puisque vous fixez vous-même le nom de fichier dans le chargeur.

Notre programme charge d'abord ce fichier, la limite de la mémoire est ensuite fixée directement en dessous de notre routine ainsi chargée, de façon à ce que notre routine ne puisse être effacée ni par des variable ni par les buffers du DOS. Ce n'est qu'ensuite que sont créés les buffers du DOS et que la limite de la mémoire est à nouveau abaissée. La routine ne peut plus maintenant être effacée. Le mieux est de toujours éteindre

puis rallumer l'ordinateur avant d'utiliser de tels programmes car il peut y avoir certaines complications si vous aviez auparavant un programme qui avait déjà mis en place le buffer du DOS ou si la limite supérieure avait déjà été abaissée d'une manière ou d'une autre. Après tout cela, le mieux est que vous définissiez encore QUATRE variables dont vous aurez en permanence besoin dans le programme.

```
ACTIVE=&9650+0
MSG=&9650+3
DESACTIVE=&9560+6
DS$=""
```

Avec une autre adresse de départ de la routine, vous devez bien sûr modifier ces valeurs. Il s'agit ici des adresses d'entrée des différentes routines. Nous vous souhaitons beaucoup de succès dans l'utilisation de ces routines. Mais avant que vous ne vous précipitiez pour taper ce chargeur Basic: dans ce chapitre figure un autre paquet de routines pour rendre possible la sauvegarde relative de fichiers. Ces routines ont été réalisées à l'aide des instructions RSX plus puissantes. La routine d'interception des erreurs est intégrée dans ce paquet de routines. Si vous voulez donc utiliser la sauvegarde relative de fichier, il vaut donc mieux que vous tapiez le chargeur Basic de la sauvegarde relative de fichier pour éviter beaucoup de travail inutile.

Les possesseurs d'un CPC 664 qui jugent la routine d'interception des erreurs inutile parce qu'elle est soutenue par leur Basic ne doivent pas perdre de vue trois faits importants:

1) Les messages d'erreur apparaissent obligatoirement sur l'écran; cela est difficilement admissible pour des programmes professionnels.

2) Vous n'avez pas de compatibilité avec les ordinateurs CPC 464.

3) L'interrogation de la variable ERR vous indique s'il y a une erreur disque mais vous n'obtenez jamais le message d'erreur précis et vous ne pouvez pas l'obtenir non plus en utilisant la variable DERR! Il ne vous est donc pas possible de distinguer un DISC FULL du message d'erreur DISC IS MISSING.

5.2.1 Explications sur la routine d'interception des erreurs

Pour intercepter les messages d'erreur certaines connaissances sur l'intérieur du CPC sont nécessaires car on ne doit pas sous-estimer l'intervention dans le système d'exploitation qu'il est nécessaire d'entreprendre. Nous nous servons pour cela de la possibilité qu'offre le CPC de DETOURNER certains vecteurs. Nous devons utiliser abondamment cette possibilité.

Lorsque le lecteur de disquette sort un message d'erreur sur l'écran, la routine du système d'exploitation TXT OUTPUT, en &BB5A, est appelée. Cette routine sert à sortir sur l'écran dans la fenêtre actuelle un caractère placé dans l'accumulateur. Cette routine est donc détournée.
La routine d'interception des erreurs est donc placée et exécutée AVANT la routine TXT OUTPUT normale, et ce pour CHAQUE caractère devant être affiché à l'écran. Cette routine examine si le caractère à afficher vient du lecteur de disquette. A cet effet, on examine l'adresse de retour placée sur la pile. La ROM disquette est située dans la zone AU DESSUS de &C000. C'est ici également que se trouve le système d'exploitation Basic. La routine de transmission des messages d'erreur est située dans la zone de &CB00 à &CBFF. Il nous suffit donc d'interroger l'octet fort de l'adresse de retour. Si donc TXT OUTPUT est appelée à partir de cette zone, il s'agit d'un caractère venant du lecteur de disquette qui est intercepté par la routine et stocké provisoirement pour une utilisation ultérieure.
Un flag est mis simultanément pour qu'on puisse ensuite tester si une erreur s'est produite. Le flag sert en outre à intercepter le message BREAK. Le message BREAK ne vient pas en effet de la ROM disquette mais de la ROM Basic normale. Tout cela est réalisé dans la routine OUTCHK (voyez le listing assembleur). Dans cette routine, les vecteurs sont détournés et ces vecteurs sont à nouveau fixés à leur valeur normale dans la routine RESET.
Lorsque le lecteur de disquette a sorti un message d'erreur, il saute à la ROM Basic, à l'endroit où les programmes sont interrompus. Le message BREAK est sorti et le programme est interrompu. Nous devons ici détourner le vecteur Ready Modus pour empêcher que les programmes puissent être interrompus. Cette routine teste si le flag d'erreur est mis car on peut fort bien parvenir au mode Ready Modus SANS qu'un message d'erreur ne se soit produit. Si le flag d'erreur est mis, le message d'erreur #18 (Unknown user function) est généré qui pourra ensuite être intercepté.

Nous avons enfin encore la routine GETTXT qui sert à nous transmettre le

message d'erreur en Basic. Nous obtenons donc le message d'erreur en clair dans une variable alphanumérique quelconque.

La routine KWC sert à envoyer un C pour répondre à la question éventuelle

Retry, Ignore or Cancel

L'erreur est ainsi immédiatement reconnue et le déroulement du programme est immédiatement interrompu.

ATTENTION! IMPORTANT!

Lorsque vous faites afficher le catalogue d'une disquette, ces caractères sont également interprétés comme une erreur car ils proviennent exactement de la même zone que les messages d'erreur. Cela vaut aussi bien pour l'instruction IDIR que pour l'instruction CAT. Lorsque la routine d'erreur est activée, l'instruction lDIR devient inopérante! Si vous voulez utiliser cette instruction, vous devez utiliser

CALL active:lDIR:CALL desactive

L'instruction CAT a été traitée ici de façon spéciale. Ses vecteurs ont également été détournés de sorte que l'instruction CAT peut être utilisée comme à la normale.

Pour tous ceux qui veulent utiliser la routine d'interception des erreurs, il est certainement également intéressant de savoir comment cette routine a été ici intégrée dans le programme. En fait 8 lignes suffisent, à partir de la ligne 3170, pour réaliser un traitement pratique des erreurs. L'ouverture d'un fichier se présente donc maintenant ainsi:

ef=0:OPENIN"nom de fichier":IF ef THEN recommencer

Le flag d'erreur ef est mis à 0 avant l'OPENIN. S'il est mis après l'OPENIN, c'est qu'une erreur s'est produite et on essaie à nouveau d'exécuter l'instruction OPENIN. Dans la routine d'erreur, l'utilisateur a la possibilité soit de revenir au menu principal, soit d'entreprendre une nouvelle tentative d'ouverture; cela peut dépendre du message d'erreur survenu. Par exemple, avec Disc missing, on peut placer la disquette dans le lecteur de disquette et essayer à nouveau (ef vaut alors 1). Pour un Read fail, cela est moins évident. Si le message

d'erreur est un File not found, la variable nf est mise à 1. Aucun texte d'erreur n'est sorti. On peut par exemple tester de cette façon si un fichier existe déjà.

```
;
;ROUTINE D'INTERCEPTION DES ERREURS
;**********************************
;
;(C) 1985 BY DATA BECKER GMBH
;
; JS 22/3/1985
;
;
        ORG  #5100
;
        JP   SET                ;FIXE POINTEUR
        JP   GETTXT             ;RETIRE CHAINE
        JP   RESET              ;RESTAURE LES VECTEURS
;
;
SET:    CALL #B900              ;ROM ENABLE
        PUSH AF                 ;RANGE ETAT ROM
        LD   A,(#DE01)          ;RETIRER CARACTERE
        LD   (VER),A            ;RANGE NUMERO DE VERSION
        POP  AF
        CALL #B90C              ;ROM DISABLE
        CP   #71                ;464 OU 664?
        JR   NZ,CPC664          ;664!
        LD   A,#C3
        LD   (#AC01),A          ;READY
CPC664: LD   A,#C3
        LD   (#BB5A),A          ;OUT CHAR
        LD   (#BB06),A          ;KM WAIT CHAR
        LD   (#BC9B),A          ;CAS CATALOG
;
        LD   A,(VER)
        CP   #71
        JR   NZ,CPC66A
        LD   HL,READYMODE
        LD   (#AC02),HL         ;DETOURNER READY
CPC66A: LD   HL,OUTCHK
        LD   (#BB5B),HL         ;DETOURNER OUT CHAR
        LD   HL,KWC
        LD   (#BB07),HL         ;DETOURNER KM WAIT CHAR
        LD   HL,CAT
        LD   (#BC9C),HL         ;CAS CATALOG
        RET
;
;
;ROUTINE TESTE, SI CARACTERE VIENT DU LECTEUR DE DISQUETTE
;
OUTCHK: EX   (SP),HL            ;RETIRE ADRESSE RETOUR
        PUSH AF                 ;SAUVE ACCU
        LD   A,H                ;TESTE OCTET FORT
        CP   #CB                ;VIENT DE LA ROM FLOPPY
        JR   NZ,NEIN            ;NON
        LD   A,1
        LD   (TESTERR),A        ;FIXER FLAG
        POP  AF                 ;NE PAS SORTIR CARACTERE
        PUSH HL                 ;SAUVE HL
        LD   HL,(HELP)          ;RETIRE POINTEUR BUFFER
        CP   10                 ;CARACTERE EST-IL LF
        JR   Z,NOT
```

```
 580 ;CR AUTORISE COMME SYMBOLE DE SEPARATION
 590         LD   (HL),A                ;MEMOIRE CARACTERE
 600         INC  HL
 610         LD   A,L
 620         CP   40                    ;40 CARACTERES?
 630         JR   NZ,N1
 640         DEC  HL                    ;40 CARACTERES MAXI
 650 N1:     LD   (HELP),HL
 660 NOT:    POP  HL
 670         PUSH AF
 680 SUPRES: POP  AF
 690         EX   (SP),HL               ;ADRESSE RETOUR
 700         RET                        ;PAS DE SORTIE
 710 ;
 720 ;
 730 NEIN:   LD   A,(TESTERR)
 740         OR   A
 750         JR   NZ,SUPRESS            ;INTERCEPTE
 760         POP  AF                    ;RETIRE CARACTERE
 770         EX   (SP),HL
 780         DEFB #CF,0,#94             ;RST 2
 790 ;
 800 ;
 810 ;
 820 ;ROUTINE POUR MODE READY
 830 ;
 840 READYM: LD   A,(TESTERR)
 850         OR   A                     ;FIXE FLAGS
 860         RET  Z                     ;PAS D'ERREUR
 870         XOR  A
 880         LD   (TESTERR),A
 890         LD   HL,BUFFER             ;ADRESSE BUFFER
 900         LD   (HELP),HL
 910         LD   E,18
 920         JP   #CA94
 930 ;
 940 ;DETOURNER #BB06 KM WAIT CHAR
 950 ;VALEUR DEFAUT POUR ERREUR ELECTRONIQUE
 960 ;C=CANCEL
 970 ;
 980 KWC:    PUSH AF                    ;SAUVE ACCU
 990         LD   A,(TESTERR)
1000         OR   A                     ;FIXE FLAGS
1010         JR   NZ,DEFAULT
1020         POP  AF                    ;PAS DEFAUT-C
1030         DEFB #CF,#3C,#9A
1040 DEFAUL: POP  AF
1050         LD   A,"C"                 ;DEFAUT
1060         RET
1070 ;
1080 ;
1090 ;
1100 ;
1110 ;
1120 ;ROUTINE POUR RESTITUER LE TEXTE DE L'ERREUR
1130 ;
1140 GETTXT: LD   E,2
1150         CP   01                    ;1 PARAMETRE ?
1160         RET  NZ                    ;MAUVAIS NOMBRE PARAMETR
1170         LD   E,(IX)                ;OCTET FAIBLE
1180         LD   D,(IX+1)              ;OCTET FORT
1190         LD   C,#FF                 ;COMPTEUR
1200         LD   HL,BUFFER
1210 L1:     LD   A,(HL)
1220         CP   13                    ;CARRIAGE RETURN?
1230         INC  HL
```

```
1240          JR   Z,L1
1250          DEC  HL                ;TROUVE PREMIER CAR.TEXT
1260          PUSH HL                ;RANGER
1270          DEC  HL                ;MOINS UN
1280 LOOP:    INC  C
1290          INC  HL
1300          LD   A,(HL)
1310          CP   13                ;CARRIAGE RETURN?
1320          JR   NZ,LOOP
1330          LD   A,C
1340          LD   (DE),A            ;RANGER LONGUEUR
1350          POP  HL                ;ADRESSE TEXTE
1360          EX   DE,HL
1370          INC  HL
1380          LD   (HL),E            ;OCTET FAIBLE
1390          INC  HL
1400          LD   (HL),D            ;OCTET FORT
1410          RET
1420 ;
1430 ;
1440 ;RESTAURER LES POINTEURS
1450 ;
1460 RESET:   LD   A,(VER)           ;NUMERO DE VERSION
1470          CP   #71
1480          JR   NZ,CPC66B
1490          LD   A,#C9
1500          LD   (#AC01),A
1510 CPC66B:  LD   A,#CF             ;RST 2
1520          LD   (#BB5A),A         ;OUT CHAR
1530          LD   (#BB06),A         ;KM WAIT CHAR
1540          LD   A,#DF             ;RST 38H
1550          LD   (#BC9B),A         ;CAS CATALOG
1560 ;
1570          LD   HL,#9400          ;OUT CHAR
1580          LD   (#BB5B),HL
1590          LD   HL,#9A3C          ;KM WAIT CHAR
1600          LD   (#BB07),HL
1610          LD   HL,#A88B
1620          LD   (#BC9C),HL        ;CAS CATALOG
1630          RET
1640 ;
1650 CAT:     PUSH AF                ;SAUVE REGISTRES
1660          PUSH HL
1670          CALL RESET             ;RESTAURE VALEURS
1680          POP  HL                ;D'ORIGINE
1690          POP  AF
1700          CALL #BC9B             ;CAS CATALOG
1710          PUSH HL
1720          PUSH AF
1730          CALL SET               ;ET ACTIVER A NOUVEAU
1740          POP  AF
1750          POP  HL                ;RESTAURE REGISTRES
1760          RET
1770 ;
1780 ;POINTEURS ET MEMOIRE AUXILIAIRE
1790 ;*******************************
1800 ;
1810 TESTER:  DEFB #0
1820 VER:     DEFB 0
1830 HELP:    DEFW BUFFER
1840 BUFFER:  DEFM "OK"
1850          DEFB #0D               ;CR
1860          DEFS 40-3              ;BUFFER 40 CARACTERES
```

```
10 '*******************************************
20 '*** Routine d'interdiction des erreurs ******
30 '*** (C) 1985 by DATA BECKER GmbH JS 22/3/85 *
40 '*******************************************
50 '
60 DEFINT a-z
70 adresse = &A000 : 'Adresse de depart de la routine
80 :
90 DATA &C3,&09,&51,&C3,&83,&51,&C3,&AC,&51,&3E,&C3,&32,&01,
&AC,&32,&5A
100 DATA &BB,&32,&06,&BB,&32,&9B,&BC,&21,&60,&51,&22,&02,&AC
,&21,&30,&51
110 DATA &22,&5B,&BB,&21,&74,&51,&22,&07,&BB,&21,&D1,&51,&22
,&9C,&BC,&C9
120 DATA &E3,&F5,&7C,&FE,&CB,&20,&1E,&3E,&01,&32,&E3,&51,&F1
,&E5,&2A,&E4
130 DATA &51,&FE,&0A,&28,&0B,&77,&23,&7D,&FE,&28,&20,&01,&2B
,&22,&E4,&51
140 DATA &E1,&F5,&F1,&E3,&C9,&3A,&E3,&51,&B7,&20,&F7,&F1,&E3
,&CF,&00,&94
150 DATA &3A,&E3,&51,&B7,&C8,&AF,&32,&E3,&51,&21,&E6,&51,&22
,&E4,&51,&1E
160 DATA &12,&C3,&94,&CA,&F5,&3A,&E3,&51,&B7,&20,&04,&F1,&CF
,&3C,&9A,&F1
170 DATA &3E,&43,&C9,&1E,&02,&FE,&01,&C0,&DD,&5E,&00,&DD,&56
,&01,&0E,&FF
180 DATA &21,&E6,&51,&7E,&FE,&0D,&23,&28,&FA,&2B,&E5,&2B,&0C
,&23,&7E,&FE
190 DATA &0D,&20,&F9,&79,&12,&E1,&EB,&23,&73,&23,&72,&C9,&3E
,&C9,&32,&01
200 DATA &AC,&3E,&CF,&32,&5A,&BB,&32,&06,&BB,&3E,&DF,&32,&9B
,&BC,&21,&00
210 DATA &94,&22,&5B,&BB,&21,&3C,&9A,&22,&07,&BB,&21,&8B,&A8
,&22,&9C,&BC
220 DATA &C9,&F5,&E5,&CD,&AC,&51,&E1,&F1,&CD,&9B,&BC,&E5,&F5
,&CD,&09,&51
230 DATA &F1,&E1,&C9,&00,&E6,&51
```

```
240 :
260 FOR i=adresse TO adresse+&E5
270  READ a
280  POKE i,a
290  s=s+a
300 NEXT
310 IF s<>28065 THEN PRINT CHR$(7)"*** Erreur en DATAS ***"
: END
320 :
330 DATA &01,&09,&04,&83,&07,&ac,&18,&60,&1e,&30,&24,&74,&2a
,&d1,&3a,&e3
340 DATA &3f,&e4,&4e,&e4,&56,&e3,&61,&e3,&67,&e3,&6a,&e6,&6d
,&e4,&76,&e3
350 DATA &91,&e6,&d4,&ac,&de,&09,&e4,&e6
360 :
370 FOR i=1 TO 20
380  READ of1,of2
390  ad1=adresse+of2
400  POKE adresse+of1,ad1 AND 255 : 'octet faible
410  POKE adresse+of1+1,INT(ad1/256) AND &FF
420 NEXT
430 :
440 INPUT "Le fichier doit-il etre sauve? (O/N) ",a$
450 IF UPPER$(a$)="O" THEN INPUT "Nom du fichier : ",b$ : SA
VE b$,b,adresse,&E8
```

5.3 GESTION DE FICHIERS RELATIFS

Vous avez appris au chapitre 1.5 comment on programme les fichiers séquentiels et vous êtes certainement impatient de découvrir les fichiers relatifs qui offrent vraiment quelques avantages remarquables. Ce mode de programmation n'est toutefois pas toujours optimal: notamment pour de petites applications, l'utilisation des fichiers relatifs ne présente aucune intérêt car elle prendrait plus de temps et de place que la programmation séquentielle des fichiers. Ce n'est que si des corrections et divers accès au fichier sont souvent nécessaires que les fichiers relatifs deviennent très vite intéressants. Outre la différence dans le travail exigé de l'utilisateur par les fichiers relatifs ou les fichiers séquentiels, il existe d'autres différences importantes.

Tout d'abord, avec un fichier relatif, il n'est plus nécessaire de charger un fichier entièrement en mémoire pour pouvoir le traiter. Les boucles compliquées pour rechercher ou pour lire un enregistrement déterminé du fichier deviennent également superflues. Il y a au lieu de cela d'autres règles qu'il faut suivre: c'est ainsi qu'il faut réfléchir auparavant très sérieusement à la taille maximale que pourra avoir un enregistrement (combien de caractères notre enregistrement pourra-t-il comporter?), et au nombre d'enregistrements que contiendra notre fichier (combien d'enregistrements voulons-nous traiter?). Avec les fichiers séquentiels, vous pouviez jusqu'ici faire l'économie de cette réflexion préalable, bien que tout programmeur sérieux doive en fait toujours essayer d'évaluer la taille de son fichier avant de commencer son travail, ne serait-ce que pour savoir combien d'enregistrements rentrent sur une disquette.

Nous allons maintenant effectuer un tel calcul en reprenant notre exemple du chapitre 1.5; vous vous souvenez certainement de notre agenda téléphonique:

Nous avons TROIS champs de données

Champ 1) Nom
Champ 2) Prénom
Champ 3) Numéro de téléphone

Supposons maintenant que vous vouliez sauvegarder 50 enregistrements de ce type, nous aurons alors 50 enregistrements.

Pour pouvoir accepter le cas échéant d'autres enregistrements, nous créerons 100 enregistrements. Il nous faut maintenant déterminer encore la taille des enregistrements. Il faut pour cela attribuer d'abord à chaque champ de données une longueur maximale qui ne pourra plus être ensuite dépassée dans le programme. Vous devez donc veiller à ce que ces limites soient respectées car sinon cela peut entraîner des complications. Il s'agit là d'une contrainte que vous retrouverez pour tous les fichiers relatifs, même sur les plus gros ordinateurs. La taille d'un enregistrement est un élément important pour un bon déroulement des calculs qui permettront de déterminer où tel ou tel enregistrement se trouve.
Avec NOTRE gestion de fichiers relatifs, vous avez même une certaine marge de manoeuvre puisque vous pouvez dépasser sans crainte les limites des différents champs de données pourvu que vous raccourcissiez le champ suivant en fonction de ce dépassement. Nous avons prévu cela pour rendre la gestion de fichier un peu plus souple mais il n'en va pas de même sur tous les systèmes! Vous devez simplement veiller à ce que la longueur totale ne soit pas dépassée. Mais revenons à notre exemple:

Nous devons attribuer maintenant les longueurs maximales, c'est-à-dire le nombre maximum de caractères autorisé pour chaque CHAMP DE DONNEES. Choisissons par exemple les limites suivantes:

Champ 1) Nom	(25 caractères)
Champ 2) Prénom	(15 caractères)
Champ 3) Numéro de téléphone	(15 caractères)
Total	55 caractères

Le nom peut donc comporter au maximum 25 caractères, ce qui devrait suffir même pour des doubles noms, le prénom est limité à 15 caractères. Nous avons enfin encore prévu au plus 15 caractères pour le numéro de téléphone. Cela nous donne au total 55 caractères par enregistrement. Il faut encore ajouter 3 à ces 55 caractères car nous avons trois champs de données qui doivent être séparés les uns des autres. Le symbole de séparation ne serait pas nécessaire si les champs de données avaient une taille fixe mais comme ce n'est pas le cas nous aurons besoin également avec les fichiers relatifs d'un symbole de séparation. Nous arrivons ainsi à une longueur de

58 caractères

Mais pour que la routine de gestion de fichiers relatifs ne soit pas trop longue -car c'est vous qui devrez la taper- nous avons prévu une limite: la longueur d'enregistrement doit être arrondie à la prochaine puissance de 2. Dans notre exemple, cela signifie que nous indiquerons 64 comme longueur d'enregistrement.

La longueur maximale d'enregistrement est de 512 octets, vous avez donc le choix entre les longueurs d'enregistrements suivantes:

2, 4, 8, 16, 32, 64, 128, 256 et 512

Vous pouvez ainsi choisir entre 9 différentes longueurs d'enregistrement possibles. Si la longueur d'enregistrement que vous avez calculée est par exemple de 32, alors vous avez de la chance et vous pouvez indiquer aussi 32 comme longueur. Si par contre vous arrivez à un caractère de plus, vous devez soit choisir la longueur d'enregistrement 64, soit essayer de réfléchir à nouveau si vous ne pouvez pas raccourcir l'un ou l'autre des champs de données. Dans notre exemple, nous avons calculé une longueur de 58. La différence entre 64 et 58 est de 6. Vous aurez donc 6 octets en trop pour chaque enregistrement qui seront ainsi "gaspillés". Vous pouvez donc prendre en compte cette longueur d'enregistrement et augmenter encore l'un ou l'autre des champs de données de 1 ou 2 caractères. On peut cependant tout à fait accepter cette contrainte. Pour les fichiers relatifs, vous pouvez oublier les instructions utilisées jusqu'ici pour les fichiers séquentiels, à part les instructions OPENIN et CLOSEIN. Par contre nous utiliserons 4 nouvelles instructions:

|INIT,<longueur>
|RECORD,<numéro d'enregistrement>
|PUT,<liste de paramètres>
|GET,<liste de paramètres>

L'instruction GET est comparable à l'instruction LINE INPUT et concerne donc la lecture des données, PUT sert à l'écriture des données et est donc comparable à PRINT.

Lorsqu'un champ de données doit comporter des nombres, ces nombres doivent être convertis en une chaîne de caractères avec l'instruction STR$. Pour les nombres réels, la longueur d'un champ de données est alors de 12 et de 9 seulement pour les nombres entiers.

Comme le lecteur de disquette DDI-1 ne permet pas normalement le travail

avec des fichiers relatifs, ceux-ci seront simulés avec des fichiers séquentiels. Avant que nous puissions travailler avec un fichier relatif, celui-ci doit d'abord avoir été "créé". Nous réalisons cela en créant un fichier séquentiel dans lequel il n'y aura rien. Souvenez-vous que notre fichier d'exemple doit comporter 100 enregistrements avec une longueur d'enregistrement de 64 caractères. Nous avons donc besoin d'une place mémoire de 100*64 = 6400 octets. La création d'un tel fichier est très simple:

```
10 REM ============================
20 REM   Création du fichier
30 REM ============================
40 :
50 OPENOUT "Fichier.Rel"
60 FOR i=0 TO 100
70  PRINT STRING$(64,32);
80 NEXT
90 CLOSEOUT
```

La boucle en ligne 60 doit contenir le nombre d'enregistrements prévu, 100 dans notre exemple. La ligne 70 réserve la place nécessaire pour chaque enregistrement; nous remplissons donc notre fichier (encore) séquentiel avec des caractères espace (vous pouvez choisir n'importe quel caractère). Le point-virgule en ligne 70 permet que le calcul tombe tout à fait juste. Ces quelques lignes sont suffisantes pour créer un fichier relatif que nous allons maintenant pouvoir traiter.

Nous pouvons maintenant écrire 100 enregistrements dans notre fichier et dorénavant chaque enregistrement aura un numéro qui le définit de façon précise. Le premier enregistrement aura le numéro d'enregistrement 0.
Lorsque vous ouvrez un fichier relatif, vous pouvez toujours utiliser l'instruction OPENIN. Il est possible d'écrire et de lire simultanément dans un fichier relatif mais l'ouverture se fait toujours avec OPENIN!!

L'instruction IINIT permet de définir la longueur d'enregistrement. Cela doit être fait IMMEDIATEMENT après l'ouverture du fichier. OPENIN et IINIT marchent toujours ensemble, comme deux frères siamois. L'instruction IINIT a un paramètre; vous pouvez utiliser aussi bien des constantes (ce qui sera le plus souvent le cas) que des variables. Dans notre exemple, vous devriez entrer:

```
IINIT,64
```

Il serait cependant également possible d'entrer:

```
longueurenregistrement=64:lINIT,longueurenregistrement
```

Avec l'instruction IRECORD, vous pouvez déterminer quel est le prochain enregistrement qui devra être traité; il est indifférent à cet égard que vous vouliez lire ou écrire cet enregistrement. Le premier enregistrement a le numéro 0. Si cela vous ennuie, vous pouvez fort bien laisser de côté ce premier enregistrement ou l'utiliser comme mémoire spéciale. L'instruction lRECORD a également un paramètre qui peut également être une constante ou une variable. Avec l'instruction IRECORD il est cependant plus probable qu'on ait à utiliser une variable comme paramètre. Pour vous positionner par exemple sur le 36ème enregistrement, entrez:

```
lRECORD,35
```

Les instructions lGET et lPUT ont une syntaxe identique. L'instruction lGET retire des données de l'enregistrement sélectionné et l'instruction lPUT écrit des données dans l'enregistrent sélectionné. Les instructions IGET et IPUT ont un nombre de paramètres quelconque compris entre 1 et 32. Si vous n'entrez aucun paramètre, vous n'aurez pas de message d'erreur, mais l'instruction restera sans effet. Vous ne pouvez transmettre QUE des variables alphanumériques. Les constantes qui doivent ètre écrites doivent auparavant avoir été affectées à une variable alphanumérique. Supposons par exemple que nous voulions écrire le mot "Cheval" dans le 24ème enregistrement. Nous devrons alors entrer la séquence d'instructions suivante:

```
10 OPENIN "Fichier.REL"
20 IINIT,64
30 IRECORD,23
40 a$="Cheval"
50 lPUT,@a$
60 IINIT,0
70 CLOSEIN
```

Cette exemple nous permet de noter deux particularités. Avant de fermer un fichier relatif, vous DEVEZ entrer l'instruction IINIT,0 et ce n'est qu'après que vous pouvez fermer le fichier. Cela garantit que des données qui pourraient se trouver encore en mémoire avant la fermeture du fichier soient écrites sur la disquette. Une seconde particularité tient à

l'instruction IPUT: la variable alphanumérique ne peut pas être indiquée directement après la virgule, vous devez placer un @ devant. Il ne s'agit pas là d'une lubie des auteurs de cet ouvrage mais d'un défaut du système d'exploitation qui ne prévoit que cette possibilité comme unique interface entre le Basic et le langage machine. Cette petite difficulté ne devrait cependant pas vous gêner outre mesure car il est facile de s'y habituer. Si vous voulez transmettre plus d'un paramètre à l'instruction IPUT, vous devez séparer les différents paramètres entre eux par des virgules. Nous allons par exemple écrire encore le mot "Chat" dans notre enregistrement:

```
10 OPENIN "Fichier.REL"
20 IINIT,64
30 IRECORD,23
40 a$="Cheval"
45 b$="Chat"
50 IPUT,@a$,@b$
60 IINIT,0
70 CLOSEIN
```

Il est par ailleurs indifférent que vous écriviez tous les champs de données d'un enregistrement dans une seule instruction IPUT ou que vous le fassiez en plusieurs instructions car des pointeurs internes veillent déjà au bon déroulement des opérations. Nous pouvons relire les données tout aussi simplement que nous les avons écrites, grâce à l'instruction IGET.

Vous pouvez ici transmettre un nombre quelconque de variables alphanumériques comme paramètres. Il est tout aussi indifférent que vous lisiez tous les champs de données dans une seule ligne ou en plusieurs. Nous allons maintenant relire notre enregistrement:

```
10 OPENIN "Fichier.REL"
20 IINIT,64
30 a$="" : b$=""
40 IRECORD,23
50 IGET,@a$,@b$
60 PRINT a$,b$
70 IINIT,0
80 CLOSEIN
```

Lorsque vous transmettez des variables alphanumériques comme paramètres à la routine IGET, celles-ci doivent avoir été déjà rencontrées au moins

une fois dans le programme, sinon l'interpréteur Basic vous enverra le message d'erreur:

Improper argument

Voilà en fait tout ce que vous devez prendre en compte. Pour ne pas allonger les routines plus que nécessaire, nous avons renoncé à la plupart des interrogations d'erreur. Vous devez donc veiller vous-même dans votre programme à ne pas fournir de numéros d'enregistrement trop importants (ce qui pourrait entraîner de gros désagréments) et vous devez également faire attention à ne pas dépasser la longueur d'enregistrement maximale car sinon vous effacerez les enregistrements suivants. Lorsque vous avez ouvert un fichier relatif, vous ne pouvez ouvrir aucun fichier de sortie avec OPENOUT; vous ne devez donc avoir d'ouvert que le fichier relatif. Cela est d'ailleurs bien compréhensible puisque dans un fichier relatif, vous pouvez simultanément lire ET écrire.
Faites attention à bien mettre en place et protéger les buffers d'entrée et de sortie au début du programme avec l'instruction:

```
OPENOUT "Dummy" : MEMORY HIMEM-1 : CLOSEOUT
```

Vous risquez sinon en effet d'avoir des difficultés lors du travail avec les instructions disquette étendues. Si vous voulez vous-même travailler avec la routine, prenez le programme d'exemple que nous vous fournissons. Vous pouvez voir d'après le programme d'exemple comment et dans quel ordre les différentes instructions doivent être utilisées.

N'utilisez pas l'instruction CAT car elle ferme le fichier sans sauver encore sur disquette des données éventuellement nécessaires et elle efface le buffer.

Si vous observez les règles, vous aurez beaucoup de réussite dans l'utilisation de cette routine. Le fichier relatif peut avoir n'importe quelles dimensions et il peut s'étendre théoriquement sur toute la surface de la disquette. Les temps d'accès sont très courts comme le démontre le programme d'exemple. Nous avons intégré également, outre les instructions pour la gestion des fichiers relatifs, les instructions pour la routine d'interception des erreurs (voir chapitre 5.2). Les instructions doivent être manipulées exactement comme décrit au chapitre 5.2. CALL active devient lAN et CALL desactive devient lAUS et CALL msg devient lERR. Vous disposez donc encore des trois instructions suivantes:

|AN
|AUS
|ERR,@<variable alphanumérique>

Ces sept nouvelles instructions remédient à toutes les insuffisances du lecteur de disquette. Vous pouvez maintenant construire des programmes pratiques et sans erreur, sans avoir à recourir à toute sorte d'astuces. Nous vous fournissons un listing assembleur ainsi qu'un chargeur Basic. Voyez au chapitre 5.2 comment calculer l'adresse à laquelle doit figurer l'extension. La longueur de cette routine est de &297 = 633 octets. Normalement (c'est-à-dire pour un programme sans l'instruction SYMBOL AFTER), vous pouvez utiliser &93E0 comme adresse de base.

Comme vous pouvez voir d'après le programme machine comment le problème a été précisément résolu, nous n'expliquerons ici que le principe suivant lequel la gestion de fichier a été réalisée.

5.3.1 EXPLICATION DU PROGRAMME MACHINE

Si nous voulons avoir un fichier relatif, celui-ci doit d'abord être "créé". Nous créons d'abord à cet effet un fichier séquentiel de la longueur voulue. Nous réservons ainsi sur la disquette la place voulue et occupons le nombre nécessaire de blocs (voyez également à ce sujet le chapitre 2). Les blocs occupés sont enregistrés dans le catalogue de la disquette, avec une entrée du catalogue par bloc de 16Koctets. Lorsque nous ouvrons maintenant notre fichier relatif, l'instruction |INIT,RL détermine quels blocs sont occupés par le fichier. Tous les numéros de bloc sont sauvegardés (1 bloc est constitué de deux secteurs consécutifs). Cette table est la base de la gestion de fichiers relatifs car on ne pourrait pas sinon accéder librement aux différents enregistrements. La longueur d'enregistrement est en outre sauvegardée séparément mais attention: la routine ne contrôle pas si la longueur d'enregistrement est vraiment une puissance de deux. Même si vous ouvrez un fichier relatif qui a la même longueur d'enregistrement qu'un autre fichier ouvert auparavant, l'instruction INIT doit ABSOLUMENT suivre immédiatement l'instruction OPENIN car sinon ce sont les anciens numéros de bloc qui restent en mémoire!

A partir des numéros de bloc on peut alors chaque fois calculer le secteur à appeler, grâce à l'instruction |RECORD. La routine est programmée de façon à ce qu'un secteur ne soit chargé que si c'est

absolument nécessaire. Il se peut en effet que ce secteur figure déjà en mémoire; cela sera notamment le cas si vous lisez tous les enregistrements du premier au dernier. Dans ce cas, notre méthode permettra donc de gagner beaucoup de temps. Il faut en outre, avant de lire un secteur, déterminer si le secteur se trouvant actuellement en mémoire a été écrit avec IPUT. Si c'est le cas, l'ancien secteur sera d'abord sauvegardé avant qu'un nouveau ne soit chargé.

Avec l'instruction IPUT, toutes les variables alphanumériques indiquées sont copiées dans le buffer de secteur mais avec l'instruction IGET les pointeurs des variables alphanumériques sont simplement détournés vers le buffer de secteur et leur longueur est modifiée.

Voilà donc brièvement le principe de base qui est réalisé dans cette routine. Nous avons opté pour les instructions RSX parce qu'elles sont un peu plus faciles à manipuler et qu'elles sont en même temps plus parlantes que des CALLs. De cette façon vous n'avez (presque) pas non plus à vous préoccuper de la zone mémoire dans laquelle est située la routine.

```
 10          ORG   #8000
 20 ;
 30          CALL #B900                ;ROM ENABLE
 40          PUSH AF
 50          LD    A,(#DE01)           ;VERSION
 60          LD    (VER),A             ;RANGER
 70          POP   AF
 80          CALL #B90C                ;RESTORE
 90          LD    DE,0                ;OFFSET
100          LD    A,(VER)
110          CP    #71
120          JR    Z,Q464              ;464, OK
121          LD    A,#DF
122          LD    (A0),A
123          LD    (A1),A
124          LD    (A2),A
125          LD    HL,TABU1            ;MULTIPLICATION
126          LD    (A0+1),HL
127          LD    HL,TABU2
128          LD    (A1+1),HL
129          LD    (A2+1),HL
130          JR    RSXON
140 Q464:    LD    HL,#BDBE
160          LD    (A0+1),HL
170          LD    HL,#BDC1
190          LD    (A1+1),HL
200          LD    (A2+1),HL
210 RSXON:   LD    BC,RSX              ;EXTENSIONS
220          LD    HL,KERNAL           ;4 OCT.RAM POUR SYSTEME
230          JP    #BCD1               ;INTEGRER EXTENSIONS
240 ;
250 RSX:     DEFW TABLE                ;ADRESSE DES MOTS INSTR.
260          JP    RECORD
270          JP    GET
280          JP    PUT
290          JP    INIT                ;DETERMINER LONGUEUR
300          JP    SET                 ;ACTIVER
310          JP    RESET               ;DESACTIVER
320          JP    GETTXT              ;RETIRE TEXTE
330 ;
340 TABLE:   DEFM "RECOR"
350          DEFB "D" + #80
360          DEFB "G","E","T"+#80
370          DEFB "P","U","T"+#80
380          DEFB "I","N","I","T"+#80
390          DEFB "A","N"+#80
400          DEFB "A","U","S"+#80
410          DEFB "E","R","R"+#80
420          DEFB 0                    ;FIN TABLE
430 ;
440 KERNAL:  DEFS 4                    ;MEMOIRE POUR KERNAL
450 ;
460 RECORD:  CP    01                  ;1 PARAMETRE
470          RET   NZ                  ;MAUVAIS NOMBRE
480          LD    D,(IX+1)            ;RETIRE ENREGISTREMENT#
490          LD    E,(IX)
500          LD    HL,(RECLEN)         ;RECORD LENGTH
510 A0:      CALL #BDBE                ;HL=HL*DE
520          LD    DE,512              ;LONGUEUR D'UN SECTEUR
530 A1:      CALL #BDC1                ;HL=HL/DE -- DE=REST
540          LD    (SECNR),HL          ;RANGE NUMERO SECTEUR
550          LD    (OFFSET),DE         ;RANGER DECALAGE
```

```
 560           XOR  A                          ;A:=0
 570           LD   (POINTER),A                ;POINTEUR SUR ZERO
 580           LD   (POINTER+1),A
 590           LD   DE,(OLDSEC)
 600           SBC  HL,DE                      ;MEME SECTEUR?
 610           RET  Z                          ;SI ZERO DANS BUFFER
 620           CALL SAVE                       ;SAUVER D'ABORD BUFFER
 630           LD   HL,(SECNR)                 ;RETIRE No SECTEUR
 640           LD   (OLDSEC),HL                ;RANGER SECTEUR
 650           RR   H                          ;/2
 660           RR   L                          ;/2
 670           PUSH AF                         ;RANGE CARRY
 680           LD   DE,(#A79B)                 ;TABLE BLOCS
 690           ADD  HL,DE                      ;RETIRE No BLOC
 700           LD   A,(HL)                     ;No BLOC
 710           LD   L,A
 720           LD   H,0                        ;HL:=No BLOC
 730           ADD  HL,HL                      ;*2
 740           POP  AF                         ;RETIRE CARRY
 750           LD   DE,0
 760           ADC  HL,DE                      ;AJOUTE CARRY
 770           LD   DE,9
 780 A2:       CALL #BDC1                      ;HL:=HL/DE --- DE:=REST
 790           INC  HL                         ;+1
 800           INC  HL                         ;+1 = TRACK
 810           LD   A,L                        ;TRACK
 820           LD   (TRACK),A                  ;RANGER
 830           LD   A,E                        ;SECTOR
 840           ADD  A,#41                      ;AJOUTER DECALAGE
 850           LD   (SECTOR),A                 ;RANGER SECTEUR
 860           LD   D,L                        ;D:=TRACK
 870           LD   C,A                        ;C:=SECTOR
 880           LD   A,(#A708)                  ;DRIVE POUR OPENIN
 890           LD   E,A
 900           LD   HL,(#A751)                 ;PNTFR POUR BUFFER INPUT
 910           DEFB #DF
 920           DEFW TAB1                       ;CALL #C666, CHARGE
 930           RET                             ;SECTEUR
 940 ;
 950 ;SAUVEGARDE DU BLOC
 960 ;
 970 SAVE:     LD   A,(REAWRI)                 ;FLAG READ/WRITE
 980           OR   A
 990           RET  Z                          ;PAS NECESSAIRE ECRIRE
1000           LD   HL,(TRACK)                 ;RETIRER TRACK/SECTOR
1010           LD   D,L                        ;D:=TRACK
1020           LD   C,H                        ;C:=SECTOR
1030           LD   A,(#A708)                  ;DRIVE POUR OPENIN
1040           LD   E,A                        ;E:=DRIVE
1050           LD   HL,(#A751)                 ;INPUT-BUFFER
1060           DEFB #DF
1070           DEFW TAB2                       ;CALL #C64E ECR.SECTEUR
1080           XOR  A                          ;ACCU:=0
1090           LD   (REAWRI),A                 ;RESTAURER FLAG
1100           RET
1110 ;
1120 GET:      LD   C,0                        ;0=GET
1130           JR   GETPUT
1140 PUT:      LD   C,1                        ;1=PUT
1150 GETPUT:   OR   A                          ;TESTE NOMBRE PARAMETRES
```

```
        RET  Z              ;=0 => ALORS FIN
        LD   B,A            ;NOMBRE DE VARIABLES($!)
        ADD  A,A            ;ACCU*2
        PUSH IX             ;IX DANS
        POP  HL             ;HL
        ADD  A,L
        LD   L,A
        LD   A,H
        ADC  A,0            ;HL + NOMBRE*2-1 =
        LD   H,A            ;POINTEUR DANS PARAMETRE
        DEC  HL
        LD   A,C            ;ORDRE#
        LD   (ORDER),A      ;RANGER
LO:     PUSH HL             ;RANGER
        LD   HL,(#A751)     ;POINTEUR POUR OPENIN
        LD   DE,(OFFSET)
        ADD  HL,DE          ;AJOUTER DECALAGE
        LD   DE,(POINTER)
        ADD  HL,DE          ;AJOUTER POINTEUR
        EX   DE,HL          ;DE:=POINTEUR DS BUFFER
        POP  HL             ;RETIRER ADRESSE
        PUSH BC
        LD   B,(HL)
        DEC  HL
        LD   C,(HL)         ;ADRESSE VARIABLE DANS BC
        DEC  HL
        PUSH HL
        PUSH DE
        PUSH BC
        LD   A,(ORDER)
        OR   A
        JR   NZ,PUTV        ;PUTer VARIABLES
        LD   B,0            ;COMPTEUR SUR 0
L1:     LD   A,(DE)         ;RETIRE CARACTERE
        CP   13             ;CR=FIN?
        JR   Z,END
        INC  B              ;AUGMENTER LONGUEUR
        JR   Z,END          ;ERREUR -- TROP LONG -->
        INC  DE             ;CHAINE VIDE
        JR   L1
END:    POP  HL
        LD   (HL),B         ;RANGER LONGUEUR
        POP  DE             ;DE:=ADRESSE DE DEPART
        INC  HL
        LD   (HL),E         ;OCTET FAIBLE
        INC  HL
        LD   (HL),D         ;OCTET FORT
        LD   HL,(POINTER)
        LD   D,0
        LD   E,B            ;E:=LONGUEUR
        INC  DE             ;+1 POUR CR
        ADD  HL,DE
        LD   (POINTER),HL   ;NOUVEAU POINTEUR
LOOP:   POP  HL             ;POINTEUR SUR TABLE
        POP  BC             ;PARAMETRES
        DJNZ LO             ;PROCHAINE VARIABLE
        RET
;
PUTV:   POP  HL             ;RETIRE ADRESSE VARIABLE
        LD   C,(HL)         ;NOMBRE CARACTERES
        LD   B,0
        INC  HL
        LD   E,(HL)         ;OCTET FAIBLE ADRESSE
        INC  HL
        LD   D,(HL)         ;ET OCTET FORT
```

```
1810          EX    DE,HL              ;HL:=STRING
1820          POP   DE                 ;RETIRE ADRESSE BUFFER
1830          LD    A,C
1840          OR    A
1850          PUSH  BC                 ;RANGE NOBMRE
1860          JR    Z,EN1              ;FIN SI CHAINE VIDE
1870          LDIR                     ;DECALE CARACT.DS BUFFER
1880 EN1:     EX    DE,HL
1890          LD    (HL),13            ;CR=SYMBOLE SEPARATION
1900          POP   DE
1910          INC   DE                 ;+1 POUR CARACTERE SEPAR
1920          LD    HL,(POINTER)
1930          ADD   HL,DE
1940          LD    (POINTER),HL       ;NOUVEAU POINTEUR
1950          LD    A,1
1960          LD    (REAWRI),A         ;RANGER QUE ECRIT
1970          JR    LOOP               ;FERMER BOUCLE
1980 ;
1990 ;INITIALISE FICHIER
2000 ;
2010 INIT:    CP    01                 ;1 PARAMETRE ?
2020          RET   NZ                 ;MAUVAIS NOMBRE PARAM.
2030          LD    H,(IX+1)
2040          LD    L,(IX)             ;DETERMINER LONGUEUR
2050          LD    A,H
2060          OR    L
2070          JP    Z,SAVE             ;INSTRUCTION INIT,0
2080          LD    (RECLEN),HL        ;RANGER
2090          LD    HL,#FFFF
2100          LD    (OLDSEC),HL        ;ANNULER FLAG
2110          LD    DE,(#A79B)         ;ADRESSE BUFFER OPENOUT
2120 L10:     LD    HL,#A719           ;TABLE BLOCS
2130          LD    B,16
2140 L11:     LD    A,(HL)
2150          OR    A
2160          RET   Z                  ;DERNIER BLOC
2170          LD    (DE),A
2180          INC   DE
2190          INC   HL
2200          DJNZ  L11                ;PROCHAIN BLOC
2210          LD    HL,(#A729)
2220          LD    BC,#80
2230          ADD   HL,BC
2240          LD    (#A729),HL
2250          LD    HL,0000
2260          LD    (#A768),HL         ;VIDER BUFFER
2270          DEFB  #DF
2280          DEFW  GETCHAR            ;CALL #CF64 DISC IN CHAR
2290          JR    L10
2300 ;
2310 RECLEN:  DEFW  64                 ;LONGUEUR ENREG.1-512
2320 SECNR:   DEFW  0                  ;SECTORNR
2330 OFFSET:  DEFW  0                  ;OFFSET DS SECTEUR
2340 REAWRI:  DEFB  0                  ;FLAG 1=ECRIT,0=LU
2350 OLDSEC:  DEFW  #FFFF              ;DERNIER SECTEUR
2360 TRACK:   DEFB  0                  ;PISTE DU SECTEUR
2370 SECTOR:  DEFB  0                  ;SECTEUR DU SECTEUR
2380 TAB1:    DEFB  #66,#C6,7          ;ADRESSE #C666 DANS ROM DISQUE
2390 TAB2:    DEFB  #4E,#C6,7          ;ECRIRE SECTEUR
2400 POINTE:  DEFW  0                  ;POINTEUR DANS BUFFER
```

```
ORDER:   DEFB 0
GETCHA:  DEFB #64,#CF,7             ;CF64 GET CHAR DE
;                                    BUFFER OPENIN
;
;ROUTINE D'INTERCEPTION DES ERREURS
;*********************************
;
;(C) 1985 BY DATA BECKER GMBH
;
; JS 22/3/1985
;
;
         JP   SET                   ;FIXE POINTEUR
         JP   GETTXT                ;RETIRE CHAINE
         JP   RESET                 ;RESTAURE VECTEURS
;
;
SET:     LD   A,(VER)
         CP   #71                   ;664?
         JR   NZ,NO1
         LD   A,#C3
         LD   (#AC01),A             ;READY
NO1:     LD   A,#C3
         LD   (#BB5A),A             ;OUT CHAR
         LD   (#BB06),A             ;KM WAIT CHAR
         LD   (#BC9B),A             ;CAS CATALOG
;
         LD   A,(VER)               ;VERSION
         CP   #71
         JR   NZ,NO3                ;664
         LD   HL,READYMODE
         LD   (#AC02),HL            ;DETOURNER READY
NO3:     LD   HL,(#BB5B)
         LD   (VEK4+1),HL
         LD   HL,OUTCHK             ;DETOURNER OUT CHAR
         LD   (#BB5B),HL            ;DETOURNER OUT CHAR
         LD   HL,(#BB07)            ;RANGER ANCIEN VECTEUR
         LD   (VEK5+1),HL
         LD   HL,KWC
         LD   (#BB07),HL            ;DETOURNER KM WAIT CHAR
         LD   HL,(#BC9C)            ;ANCIEN VECTEUR
         LD   (VEK3),HL
         LD   HL,CAT
         LD   (#BC9C),HL            ;CAS CATALOG
         RET
;
;
;ROUTINE TESTE, SI CARACTERE VIENT DU LECTEUR DE DISQUETTE
;
OUTCHK:  EX   (SP),HL               ;RETIRE ADRESSE RETOUR
         PUSH AF                    ;SAUVE ACCU
         LD   A,H                   ;TESTE OCTET FORT
         CP   #CB                   ;VIENT DE ROM DISQUE ?
         JR   NZ,NEIN               ;NON
         LD   A,1
         LD   (TESTERR),A           ;FIXER FLAG
         POP  AF                    ;NE PAS SORTIR CARACTERE
         PUSH HL                    ;SAUVE HL
         LD   HL,(HELP)             ;RETIRE POINTEUR BUFFER
         CP   10                    ;CARACTERE EST LF
```

```
3010            JR    Z,NOT
3020 ;CR AUTORISE COMME SYMBOLE DE SEPARATION
3030            LD    (HL),A                 ;MEMOIRE CARACTERE
3040            INC   HL
3050            LD    A,L
3060            CP    40+BUFFER&#FF          ;40 CARACTERES?
3070            JR    NZ,N1
3080            DEC   HL                     ;40 CARACTERES MAXI
3090 N1:        LD    (HELP),HL
3100 NOT:       POP   HL
3110            PUSH  AF
3120 SUPRES:    POP   AF
3130            EX    (SP),HL                ;ADRESSE RETOUR
3140            RET                          ;PAS DE SORTIE
3150 ;
3160 ;
3170 NEIN:      LD    A,(VER)
3180            CP    #71
3190            JR    Z,NEINALT
3200            LD    A,(TESTERR)
3210            OR    A
3220            JR    Z,N12
3230            DEC   A
3240 N12:       LD    (TESTERR),A
3250            JR    NZ,SUPRESS
3260            PUSH  HL
3270            LD    HL,BUFFER
3280            LD    (HELP),HL
3290            POP   HL
3300            POP   AF                     ;RETIRE CARACTERE
3310            EX    (SP),HL
3320 VEK4:      DEFB  #CF,0,#94              ;RST 2
3330 ;
3340 NEINAL:    LD    A,(TESTERR)
3350            OR    A
3360            JR    NZ,SUPRESS
3370            POP   AF                     ;RETIRE CARACTERE
3380            EX    (SP),HL
3390            DEFB  #CF,0,#94              ;RST 2
3420 ;
3430 ;
3440 ;ROUTINE POUR MODE READY
3450 ;
3460 READYM:    LD    A,(TESTERR)
3470            OR    A                      ;FIXE FLAGS
3480            RET   Z                      ;PAS D'ERREUR
3490            XOR   A
3500            LD    (TESTERR),A
3501            LD    HL,BUFFER
3502            LD    (HELP),HL
3510            LD    E,18
3520            JP    #CA94
3530 ;
3540 ;DETOURNEMENT POUR #BB06 KM WAIT CHAR
3550 ;VALEUR DEFAUT POUR ERREUR ELECTRONIQUE
3560 ;C=CANCEL
3570 ;
3580 KWC:       PUSH  AF                     ;SAUVE ACCU
```

```
            LD    A,(TESTERR)
            OR    A                       ;FIXE FLAGS
            JR    NZ,DEFAULT
            POP   AF                      ;PAS DEFAUT-C
VEK5:       DEFB  #CF,#3C,#9A
DEFAUL:     LD    A,4
            LD    (TESTERR),A
            LD    A,"C"                   ;DEFAUT
            RET
;
;
;
;
;
;ROUTINE POUR RESTITUER TEXTE ERREUR
;
GETTXT:     LD    E,2
            CP    01                      ;1 PARAMETRE ?
            RET   NZ                      ;MAUVAIS NOMBRE PARAM.
            LD    E,(IX)                  ;OCTET FAIBLE
            LD    D,(IX+1)                ;OCTET FORT
            LD    C,#FF                   ;COMPTEUR
            LD    HL,BUFFER
LA:         LD    A,(HL)
            CP    13                      ;CARRIAGE RETURN?
            INC   HL
            JR    Z,LA
            DEC   HL                      ;TROUVE 1ER CARACT.TEXTE
            PUSH  HL                      ;RANGER
            DEC   HL                      ;MOINS UN
SLOOP:      INC   C
            INC   HL
            LD    A,(HL)
            CP    13                      ;CARRIAGE RETURN?
            JR    NZ,SLOOP
            LD    A,C
            LD    (DE),A                  ;RANGER LONGUEUR
            POP   HL                      ;ADRESSE TEXTE
            EX    DE,HL
            INC   HL
            LD    (HL),E                  ;OCTET FAIBLE
            INC   HL
            LD    (HL),D                  ;OCTET FORT
            RET
```

```
4030 ;
4040 ;
4050 ;RESTAURER LES POINTEURS
4060 ;
4070 RESET:  LD   A,(VER)             ;NUMERO DE VERSION
4080         CP   #71
4090         JR   NZ,NO2
4100         LD   A,#C9
4110         LD   (#AC01),A
4120 NO2:    LD   A,#CF               ;RST 2
4130         LD   (#BB5A),A           ;OUT CHAR
4140         LD   (#BB06),A           ;KM WAIT CHAR
4150         LD   A,#DF               ;RST 38H
4160         LD   (#BC9B),A           ;CAS CATALOG
4170 ;
4180         LD   HL,(VEK4+1)         ;OUT CHAR
4190         LD   (#BB5B),HL
4200         LD   HL,(VEK5+1)         ;KM WAIT CHAR
4210         LD   (#BB07),HL
4220         LD   HL,#A88B
4230         LD   (#BC9C),HL          ;CAS CATALOG
4240         RET
4250 ;
4260 CAT:    PUSH AF                  ;SAUVE REGISTRES
4270         PUSH HL
4280         CALL RESET               ;RESTAURE VALEURS ORIGIN
4290         POP  HL
4300         POP  AF
4310         CALL #BC9B               ;CAS CATALOG
4320         PUSH HL
4330         PUSH AF
4340         CALL SET                 ;ET RECONNECTER
4350         POP  AF
4360         POP  HL                  ;RESTAURE REGISTRES
4370         RET
4380 ;
4390 ;POINTEURS ET MEMOIRE AUXILIAIRE
4400 ;*******************************
4410 ;
4420 VEK3:   DEFW 0                   ;VECTEUR OUTCHAR
4430 TESTER: DEFB #0
4440 VER:    DEFB 0
4450 HELP:   DEFW BUFFER
4455 TABU1:  DEFB #77,#DD,#FD
4456 TABU2:  DEFB #B0,#DD,#FD
4460 BUFFER: DEFM "OK"
4470         DEFB #0D                 ;CR
4480         DEFS 40-3                ;BUFFER 40 CARACTERES
```

```
10 '********************************************
20 '*** Chargeur BASIC pour instructions RSX ***
30 '*** JS 5/5/1985 (c) by DATA BECKER GmbH ****
40 '********************************************
50 '
60 DEFINT b-z : DEFREAL s,i
70 adresse = &8000 'Adresse de depart de la routine
80 '
90 DATA CD,00,B9,F5,3A,01,DE,32,05,83,F1,CD,0C,B9,11,00
100 DATA 00,3A,05,83,FE,71,28,1C,3E,DF,32,8C,80,32,92,80
110 DATA 32,CB,80,21,08,83,22,8D,80,21,0B,83,22,93,80,22
120 DATA CC,80,18,0F,21,BE,BD,22,8D,80,21,C1,BD,22,93,80
130 DATA 22,CC,80,01,4C,80,21,7C,80,C3,D1,BC,63,80,C3,80
140 DATA 80,C3,00,81,C3,04,81,C3,7E,81,C3,DB,81,C3,C4,82
150 DATA C3,9B,82,52,45,43,4F,52,C4,47,45,D4,50,55,D4,49
160 DATA 4E,49,D4,41,CE,41,55,D3,45,52,D2,00,FC,A6,4C,80
170 DATA FE,01,C0,DD,56,01,DD,5E,00,2A,BB,81,CD,BE,BD,11
180 DATA 00,02,CD,C1,BD,22,BD,81,ED,53,BF,81,AF,32,CC,81
190 DATA 32,CD,81,ED,5B,C2,81,ED,52,C8,CD,E7,80,2A,BD,81
200 DATA 22,C2,81,CB,1C,CB,1D,F5,ED,5B,9B,A7,19,7E,6F,26
210 DATA 00,29,F1,11,00,00,ED,5A,11,09,00,CD,C1,BD,23,23
220 DATA 7D,32,C4,81,7B,C6,41,32,C5,81,55,4F,3A,08,A7,5F
230 DATA 2A,51,A7,DF,C6,81,C9,3A,C1,81,B7,C8,2A,C4,81,55
240 DATA 4C,3A,08,A7,5F,2A,51,A7,DF,C9,81,AF,32,C1,81,C9
250 DATA 0E,00,18,02,0E,01,B7,C8,47,87,DD,E5,E1,85,6F,7C
260 DATA CE,00,67,2B,79,32,CE,81,E5,2A,51,A7,ED,5B,BF,81
270 DATA 19,ED,5B,CC,81,19,EB,E1,C5,46,2B,4E,2B,E5,D5,C5
280 DATA 3A,CE,81,B7,20,24,06,00,1A,FE,0D,28,06,04,28,03
290 DATA 13,18,F5,E1,70,D1,23,73,23,72,2A,CC,81,16,00,58
300 DATA 13,19,22,CC,81,E1,C1,10,BF,C9,E1,4E,06,00,23,5E
310 DATA 23,56,EB,D1,79,B7,C5,28,02,ED,B0,EB,36,0D,D1,13
320 DATA 2A,CC,81,19,22,CC,81,3E,01,32,C1,81,18,D7,FE,01
330 DATA C0,DD,66,01,DD,6E,00,7C,B5,CA,E7,80,22,BB,81,21
340 DATA FF,FF,22,C2,81,ED,5B,9B,A7,21,19,A7,06,10,7E,B7
350 DATA C8,12,13,23,10,F8,2A,29,A7,01,80,00,09,22,29,A7
360 DATA 21,00,00,22,68,A7,DF,CF,81,18,DE,40,00,00,00,00
370 DATA 00,00,FF,FF,00,00,66,C6,07,4E,C6,07,00,00,00,64
380 DATA CF,07,C3,DB,81,C3,9B,82,C3,C4,82,3A,05,83,FE,71
390 DATA 20,05,3E,C3,32,01,AC,3E,C3,32,5A,BB,32,06,BB,32
400 DATA 9B,BC,3A,05,83,FE,71,20,06,21,74,82,22,02,AC,2A
```

```
410 DATA 5B,BB,22,67,82,21,24,82,22,5B,BB,2A,07,BB,22,91
420 DATA 82,21,88,82,22,07,BB,2A,9C,BC,22,02,83,21,F0,82
430 DATA 22,9C,BC,C9,E3,F5,7C,FE,CB,20,1E,3E,01,32,04,83
440 DATA F1,E5,2A,06,83,FE,0A,28,0B,77,23,7D,FE,36,20,01
450 DATA 2B,22,06,83,E1,F5,F1,E3,C9,3A,05,83,FE,71,28,19
460 DATA 3A,04,83,B7,28,01,3D,32,04,83,20,EA,E5,21,0E,83
470 DATA 22,06,83,E1,F1,E3,CF,00,94,3A,04,83,B7,20,D7,F1
480 DATA E3,CF,00,94,3A,04,83,B7,C8,AF,32,04,83,21,0E,83
490 DATA 22,06,83,1E,12,C3,94,CA,F5,3A,04,83,B7,20,04,F1
500 DATA CF,3C,9A,3E,04,32,04,83,3E,43,C9,1E,02,FE,01,C0
510 DATA DD,5E,00,DD,56,01,0E,FF,21,0F,83,7E,FE,0D,23,28
520 DATA FA,2B,E5,2B,0C,23,7E,FE,0D,20,F9,79,12,E1,EB,23
530 DATA 73,23,72,C9,3A,05,83,FE,71,20,05,3E,C9,32,01,AC
540 DATA 3E,CF,32,5A,BB,32,06,BB,3E,DF,32,9B,BC,2A,67,82
550 DATA 22,5B,BB,2A,91,82,22,07,BB,21,8B,A8,22,9C,BC,C9
560 DATA F5,E5,CD,C4,82,E1,F1,CD,9B,BC,E5,F5,CD,DB,81,F1
570 DATA E1,C9,8B,A8,00,71,0F,83,77,DD,FD,B0,DD,FD,0D,4C
580 DATA 20,20,20,20,20,20,20,2E,20,20,20,20,6F,6F,74,20
590 DATA 66,6F,75,6E,64,0D
600 :
610 FOR i=0 TO &325
620   READ d$
630   POKE i+adresse, VAL("&"+D$)
640   s=s+VAL("&"+D$)
650 NEXT
660 IF s<>89911 THEN PRINT CHR$(7)"*** Erreur en Datas !! **
*" : END
670 :
680 'Adapter a la zone memoire
690 :
700 FOR i=adresse TO adresse+&326
710   p=PEEK(i)
720   IF p>&7F AND p<&84 THEN 780
730 NEXT
740 PRINT CHR$(7) : INPUT"Nom du fichier :",a$
750 SAVE a$,b,adresse,&329
760 END
770 :
780 IF PEEK(I+1)>&7F AND PEEK(I+1)<&84 THEN 730
790 ad=PEEK(i-1) + p*256 : ad=UNT(ad) - &8000 + adresse
800 POKE i-1,&FF AND UNT(ad)   'octet faible
810 POKE i,&FF AND INT(ad/256) 'octet fort
820 GOTO 730
```

```
10 '*********************************
20 '  DEMONSTRATION FICHIERS RELATIFS
30 '*********************************
40 '
50 MODE 2
60 '
70 'RESPECTEZ L'ORDRE DES INSTRUCTIONS SUIVANTES Y COMPRIS
80 'DANS VOS PROPRES PROGRAMMES !!
90 '
100 OPENOUT"DUMMY":MEMORY HIMEM-1:CLOSEOUT
110 MEMORY &7FFF 'PROTEGER ROUTINE
120 'NOTRE ROUTINE D'EXTENSION DOIT A CET ENDROIT
130 'SOIT SE TROUVER DEJA EN MEMOIRE, EN &8000
140 'SOIT ETRE GENEREE EN &8000
150 'ET ETRE SAUVEGARDEE SUR DISQUETTE SOUS LE NOM
160 '"DISK.BIN". LA LIGNE SUIVANTE SERA ALORS:
170 LOAD"0:RSX1"
180 CALL &8000 'ACTIVER EXTENSIONS
190 '
200 PRINT"CREER FICHIER"
```

```
210 OPENOUT"FICHIER.REL"
220 FOR I=1 TO 600 '600 ENREGISTREMENTS
230   PRINT#9,STRING$(32,32); 'DE 32 CARACTERES
240 NEXT
250 CLOSEOUT 'FICHIER CREE
260 '
270 PRINT"ECRIRE DONNEES"
280 OPENIN"FICHIER.REL" 'OUVRIR FICHIER EN LECTURE ET ECRITU
RE
290 |INIT,32 'INITIALISE FICHIER REL
300 FOR I=0 TO 599
310   |RECORD,I
320   A$=STR$(I):B$=STR$(I^2):C$=STR$(SQR(I))
330   |PUT,@A$,@B$,@C$ 'SAUVE DONNEES
340 NEXT
350 '
360 'LIRE MAINTENANT LES 600 VALEURS DE FACON SEQUENTIELLE
370 '
380 PRINT"LECTURE SEQUENTIELLE"
```

```
390 FOR I=0 TO 599
400   !RECORD,I
410   !GET,@A$,@B$,@C$
420   PRINT I;A$;" ";B$;" ";C$
430 NEXT
440 CLOSEIN
450 '
460 'TIRER MAINTENANT 100 FOIS UN ENREGISTREMENT AU HASARD
470 '
480 OPENIN"FICHIER.REL"
490 !INIT,32
500 PRINT"TIRER AU HASARD"
510 FOR I=1 TO 100
520   X=RND*600-1
530   !RECORD,X
540   !GET,@A$,@B$,@C$
550   PRINT I;A$;" ";B$;" ";C$
560 NEXT
570 CLOSEIN
```

5.4 LE PROGRAMME DE GESTION DE FICHIER

Le chaos règne dans votre collection de disques ou vous ne savez plus dans quel classeur se trouve le plus précieux timbre de votre collection? Alors ce programme de gestion de fichier est fait pour vous.

Le programme de gestion de fichier que nous vous fournissons est relativement long mais cela ne doit pas vous décourager de le taper car il vous offre des possibilités insoupçonnées. Il n'est pas aussi puissant que DATAMAT mais il suffira certainement de par sa vitesse de travail et sa souplesse d'utilisation pour de nombreuses applications.
Peu importe donc que vous vouliez gérer vos adresses ou vos disques car aucun masque pré-défini d'entrée ne vous est imposé. Vous disposez de jusqu'à 10 champs de données pour vos enregistrements et vous pouvez occuper ces champs de données à votre guise. Le programme est conçu de sorte que vous puissiez gérer jusqu'à 200 enregistrements. Les données ne sont pas sauvegardées de façon relative mais de façon séquentielle, toutes les données étant chargées en mémoire. Nous avons opté pour cette solution entre autre par égard pour le lecteur/utilisateur car le programme aurait été sinon encore plus long.

Nous avons intégré les accents français mais ceux-ci ne participent pas correctement au tri(!) car le travail fourni serait sinon sans rapport avec l'effet recherché. Plusieurs touches ont été échangées. Si ce nouveau clavier vous gêne, vous pouvez supprimer les lignes 350-440. Il est nécessaire que la routine d'interception des erreurs du chapitre 5.2 figure également sur la disquette sur laquelle vous sauvegarderez le programme de gestion de fichier. Générez cette routine à partir de l'adresse &AOBO, comme nous vous l'expliquons au chapitre 5.2 et sauvegardez cette routine sous le nom "error.bin". C'est tout ce que vous avez à faire.

Après que le programme ait été tapé et que vous l'ayez sauvegardé au moins une fois sur dsquette (mieux vaut le sauvegarder plusieurs fois sur plusieurs disquettes) ainsi que le fichier error.bin, vous pouvez lancer le programme. Vous obtenez tout d'abord le menu à l'écran. Vous pouvez maintenant sélectionner les différents points du menu à l'aide des touches curseur, en validant votre choix avec la touche ENTER; c'est alors le point du menu apparaissant en vidéo inversée qui sera exécuté.
Les données sont sauvegardées sous USER 1 de sorte que les données sont nettement séparées des programmes et données normaux. Pour examiner la liste de vos fichiers, vous pouvez sélectionner le point du menu Afficher

contenu disquette. Seuls les fichiers sous USER 1 vous seront alors montrés, normalement donc vos fichiers. Faites-le et vous verrez que la place libre sur la disquette est également indiquée. En actionnant une touche quelconque, vous êtes alors ramené au menu principal.

Créons comme exemple un fichier que nous avons déjà pris comme exemple au chapitre 5.3. Nous créerons un agenda téléphonique avec trois champs de données:

Nom
Prénom
Téléphone

Nous pouvons facilement augmenter encore ce fichier de trois autres champs de données en ajoutant par exemple les champs de données:

Code postal
Localité
Rue

Sélectionnez le point du menu Créer/modifier masque et créons notre masque avec six champs de données. Les entrées sont validées comme toujours avec ENTER. Lorsque vous voulez modifier un champ de données, vous pouvez déplacer le curseur vers le haut avec la touche curseur haut et réécrire sur le champ de données correspondant. La correction de lettres isolées n'est pas non plus possible ici, pour des raisons tenant au système d'exploitation, vous devez donc réécrire entièrement le champ correspondant. Le curseur saute toujours, après l'entrée, au champ suivant immédiatement le dernier champ. Lorsque vous voulez quitter le point du menu Créer/modifier masque, appuyez simultanément sur les touches CTRL et ENTER. Vous vous retrouverez instantanément dans le menu principal. Dès que vous avez entré un enregistrement, vous ne pouvez que modifier le masque, mais pas ajouter de données supplémentaires!

Après que vous ayez créé le masque, vous devez choisir le point du menu Entrer/modifier données. Vous obtenez alors le masque d'entrée en mode 80 colonnes. On vous indique en haut à droite la longueur maximale de vos champs de données pour qu'une ligne ne soit pas dépassée.
Le champ dans lequel vous êtes en train d'écrire est systématiquement affiché en vidéo inversée. Entrez d'abord vos propres données. Lorsque vous aurez fini, le masque d'entrée sera à nouveau vidé et vous verrez en bas à gauche qu'un enregistrement a été sauvegardé.

Vous vous êtes peut-être déjà étonné de découvrir en bas à droite de l'écran un affichage d'état. Vous en concluez logiquement que ce qui ne doit pas être n'est pas possible et qu'il doit donc y avoir encore un autre état. Il s'agit ici de l'état de correction. Vous voyez dans la fenêtre du milieu que le fait d'actionner simultanément les touches ENTER et CTRL entraîne une modification de l'état d'erreur. Essayez donc.

Ce n'est déjà plus le mot Entrée que vous lisez sous le mot "Etat" mais Correction. Vous voyez d'autre part apparaître:

** Remplissez le masque avec le critère de recherche **

Cela a la signification suivante: lorsque vous voulez modifier un enregistrement, vous devez indiquer au programme de quel enregistrement il s'agit. Vous disposez pour cela du masque d'entrée tout entier; mais ne craignez rien, vous n'êtes pas obligé de remplir le masque entier de données. Entrez par exemple votre prénom dans le champ "Prénom", vous pouvez faire défiler les autres champs simplement avc la touche ENTER.
Dès que vous avez terminé tous les champs avec ENTER, vous vous trouvez à nouveau dans le mode d'entrée. Mais s'il y a par exemple dans votre fichier plus d'une personne portant le même prénom, il se peut qu'un autre enregistrement vous soit proposé à la correction car seul le premier enregistrement correspondant à votre description a été pris en compte.
Supposez qu'il y ait parmi vos amis deux Pierre et que votre femme vous explique que Pierre a maintenant un nouveau numéro de téléphone. De quel Pierre s'agit-il donc? Si on vous indique cependant l'ancien numéro que vous connaissiez par coeur, vous pouvez identifier sans peine le "bon" Pierre. De même que vous avez besoin dans un tel cas de plus d'informations, de même vous faut-il fournir également plus d'informations à votre ordinateur de façon à exclure toute confusion. Dans notre exemple, il serait donc plus sensé d'entrer également le numéro de téléphone dans le champ de données prévu à cet effet pour éviter ainsi tout problème. C'est ce qu'on appelle une COMPARAISON ET qui est alors exécutée, c'est-à-dire que toutes les indications fournies doivent correspondre. Si vous faites rechercher un enregistrement qui n'existe pas, vous obtenez le message d'erreur:

** Cet enregistrement n'existe pas **

Vous vous trouvez alors toujours en mode de correction et vous pouvez lancer une autre tentative.

Vous pouvez sortir du mode de correction à tout moment avec CTRL/ENTER; vous revenez alors au mode d'entrée et la correction n'est pas entreprise!!
Vous pouvez sortir à tout moment du point de menu Entrée/correction en entrant fin dans n'importe quel champ de données.

Mais venons-en maintenant brièvement aux autres points du menu:

Charger données et Sauver données
Les données entrées ainsi que le masque sont chargées ou sauvées. Vous pouvez indiquer un nom de fichier qui ne doit pas comporter plus de 8 caractères. Vous revenez au menu principal avec CTRL/ENTER.

Supprimer données
Vous pouvez supprimer un enregistrement. A cet effet, vous pouvez le faire d'abord rechercher comme dans le point du programme Créer/modifier; avant qu'il ne soit supprimé, vous devez confirmer votre intention après une question de sécurité.

Sortir données
Vous pouvez choisir si la sortie doit s'effectuer sur l'imprimante (I) ou sur l'écran (E). Après que vous ayez actionné soit la touche "E" soit la touche "I", vous pouvez sélectionner quels champs de données doivent être sortis; c'est intéressant pour sortir des listes qui ne doivent pas contenir tous les champs de données. Si un champ de données ne doit pas être sorti, appuyez simplement sur la touche ENTER pour le champ correspondant, sinon marquez le champ avec un caractère quelconque; il suffit donc que vous appuyiez ESPACE/ENTER pour que le champ soit sorti.

Afficher contenu disquette
Le catalogue de la disquette est affiché sur l'écran.

Fin du programme
Ne choisissez ce point du programme qu'après que vous ayez déjà sauvegardé sur disquette vos données et éventuellement les corrections que vous y avez apportées; le programme ne vous demande pas de confirmation de sécurité.

Tout supprimer
Toutes les données en mémoire sont supprimées, le programme est à nouveau démarré.

```
10 '****************************************************
20 '***** (c) 1985 by DATA BECKER **********************
30 '***** Auteur : Joerg Schieb ************************
40 '***** Gestion de fichier ***************************
50 '****************************************************
60 '
70 ON ERROR GOTO 3170
80 SYMBOL AFTER 91
90 MEMORY &A0B0-1
100 LOAD "0:error.bin"
110 CLEAR
120 DEFINT a-z : active=&A0B0 : msg=&A0B3 : desactive=&A0B6
: ds$="*" : CALL active
130 ON ERROR GOTO 3170
140 ON BREAK GOSUB 960
150 OPENOUT "dumm"
160  MEMORY HIMEM-1
170 CLOSEOUT
180 DIM ma$(20),m$(9),l(20),d$(200,9),da$(9)
190 cursorup$=CHR$(240):cursordown$=CHR$(241)
200 |USER,1
210 :
220 REM =======================================
230 REM     Definition des accents francais
240 REM =======================================
250 :
260 DATA 135,91,136,92,137,93,132,123,133,124,134,125,129,12
6
```

```
270 FOR i=1 TO 7:READ a,b:KEY a,CHR$(b):NEXT
280 SYMBOL 126 , 0 , 0 , 60 , 102 , 96 , 102 , 60 , 24
285 SYMBOL 123 , 96 , 48 , 120 , 12 ,  124 , 204 , 118 , 0
290 SYMBOL 124 , 60 , 102 , 60 , 102 ,  102 , 102 , 60 , 0
295 SYMBOL 125 , 48 , 24 , 102 , 102 ,  102 , 102 , 63 , 0
300 SYMBOL 91 , 48 , 24 , 60 , 102 ,    126 , 96 , 60 , 0
305 SYMBOL 92 , 12 , 24 , 60 , 102 ,    126 , 96 , 60 , 0
310 SYMBOL 93 , 60 , 102 , 60 , 102 ,   126 , 96 , 60 , 0
315 SYMBOL 94 , 120 , 204 , 120 , 12 ,  124 , 204 , 118 , 0
320 SYMBOL 95 , 60 , 102 , 0 , 102 ,    102 , 102 , 63 , 0
325 SYMBOL 96 , 60 , 102 , 24 , 24 ,    24 , 24 , 60 , 0
350 KEY DEF 29,1,124,92
360 KEY DEF 28,1,123,91
370 KEY DEF 26,1,125,93,124
380 KEY DEF 24,1,126,94,ASC("#")
390 KEY DEF 19,1,58,42
400 KEY DEF 17,1,59,43,64
410 KEY DEF 18,0,13,13,140
420 KEY 140,"*ESC*"+CHR$(13)
430 KEY DEF 71,1,ASC("y"),ASC("Y")
440 KEY DEF 43,1,ASC("z"),ASC("Z")
450 :
460 DATA Charger donn\es,Sauver donn\es,Entrer/modifier donn
\es,Supprimer donn\es,Trier donn\es,Sortir donn\es,Afficher
contenu disquette,Cr\er/modifier masque,Fin du programme,Tou
t supprimer,fin
470 :
480 i=0 : WHILE ma$(i)<>"fin" : i=i+1
490 READ ma$(i) : l(i)=LEN(ma$(i))/2
500 WEND : nmbmask = i-1
510 STAT=0:INK 0,0 : INK 1,13 : BORDER 0 : PAPER 0 : MODE 1
: PEN 1 : PEN #1,0 : PAPER #1,1
520 WINDOW #1,1,40,1,3:CLS #1:LOCATE #1,16,2:PRINT #1,"M E N
 U"
```

```
530 FOR i=1 TO nmbmask
540  LOCATE 20-l(i),5+i*2 : PRINT ma$(i)
550 NEXT
560 champ=1 : PEN #1,1
570 PAPER #1,1 : PEN #1,0
580 WINDOW #1,20-l(champ),20+l(champ),5+champ*2,5+champ*2 :
PRINT #1,ma$(champ)
590 d$=INKEY$:IF d$="" THEN 590
600 PAPER #1,0:PEN #1,1:IF d$=cursorup$ THEN PRINT #1,ma$(ch
amp):champ=champ-1:IF champ=0 THEN champ=nmbmask:GOTO 570 EL
SE 570
610 IF d$=cursordown$ THEN PRINT #1,ma$(champ):champ=champ+1
:IF champ>nmbmask THEN champ=1:GOTO 570 ELSE 570
620 IF d$<>CHR$(13) THEN 590
630 ON champ GOSUB 1680,1240,2190,2690,2810,2940,1940,2020,1
880,640:GOTO 510
640 RUN 120 'Supprime tout
650 :
660 REM ====================================
670 REM    Cherche le tableau de cha`nes da$
680 REM ====================================
690 :
700 IF nombre=0 THEN found=0 : RETURN
710 FOR i=1 TO nombre
720   FOR i9=0 TO nombrechamps-1
730     IF da$(i9)="" THEN 750
740     IF da$(i9)<>d$(i,i9) THEN 770
750   NEXT i9
760   found=i : RETURN
770 NEXT i
780 found=0 : RETURN
```

```
790 :
800 REM ====================================
810 REM   Trier tableau d'apr[s No champ
820 REM ====================================
830 :
840 IF nombre<2 THEN RETURN
850 EVERY 30,0 GOSUB 2870
860 FOR i1=1 TO nombre-1
870   ver=i1
880     FOR i2=i1+1 TO nombre
890       IF d$(i2,champ)<d$(ver,champ) THEN ver=i2
900     NEXT
910   FOR i2=0 TO nombrechamps-1
920     d$=d$(ver,i2) : d$(ver,i2)=d$(i1,i2) : d$(i1,i2)=d$
930   NEXT
940 NEXT
950 i=REMAIN(0)
960 RETURN
970 :
980 REM ====================================
990 REM Tester si un nom de fichier existe
1000 REM ====================================
1010 :
1020 nf=0 : ef=0 : OPENIN file$+".dat" : IF ef THEN 1020
1030 found=1-nf : '1=trouv\, 0=pas trouv\
1040 CLOSEIN
1050 RETURN
1060 :
1070 REM ===========================
1080 REM    Supprimer cha`ne Nr. Sn
1090 REM ===========================
1100 :
1110 IF nombre=1 THEN nombre=0 : RETURN
1120 FOR i=sn TO nombre-1
1130   FOR i1=0 TO nombrechamps-1
1140     d$(i,i1)=d$(i+1,i1)
1150   NEXT
1160 NEXT
1170 nombre=nombre-1
1180 RETURN
```

```
1190 :
1200 REM ============================
1210 REM   Sauve fichier sur disque
1220 REM ============================
1230 :
1240 GOSUB 1490 : IF file$="*ESC*" THEN RETURN ELSE GOSUB 10
20 : IF found=0 THEN 1310 ELSE WINDOW #1,1,40,25,25 : PAPER
2 : INK 2,1 : CLS #1 : PRINT #1,CHR$(7)"Doit ]tre remplac\ (
o,n) "
1250 EVERY 20,0 GOSUB 1290
1260 d$=INKEY$ : IF d$="" THEN 1260
1270 im=REMAIN(0):IF UPPER$(d$)<>"O" THEN 1240
1280 GOTO 1310
1290 LOCATE #1,30,1 : IF im=0 THEN PRINT#1,"?";:im=1 ELSE PR
INT#1," ";:im=0
1300 RETURN
1310 ef=0 : OPENOUT file$+".dat" : IF ef THEN 1310
1320  PRINT#9,nombrechamps
1330 FOR i=0 TO nombrechamps-1
1340  PRINT #9,m$(i)
1350 NEXT
1360 PRINT#9,nombre
1370  FOR i=1 TO nombre
1380   FOR i1=0 TO nombrechamps-1
1390    PRINT#9,d$(i,i1)
1400   NEXT
1410  NEXT
1420 CLOSEOUT
1430 RETURN
1440 :
1450 REM ==============================
1460 REM   Entr\e du nom de fichier
1470 REM ==============================
1480 :
1490 MODE 1 : WINDOW #1,1,40,1,3 : PAPER #1,2 : INK 2,1 : CL
S#1 : LOCATE #1,7,2 : PRINT #1,"Entrez le nom de fichier:"
1500 WINDOW #1,1,40,15,16 : CLS #1 : PRINT#1,TAB(10)"N'utili
sez pas (,.:"CHR$(34)")
1510 PRINT#1,TAB(8)"Entrez au maximum 8 caract[res
1520 WINDOW #1,1,40,6,10 : CLS #1
1530 LOCATE #1,5,3 : INK 3,3
1540 PRINT #1,"Nom de fichier :";
1550 WINDOW #1,21,29,8,8 : PAPER #1,3 : CLS #1
1560 LINE INPUT #1,"",file$ : IF file$="*ESC*" THEN RETURN
1570 IF LEN(file$)>8 THEN PEN 3:LOCATE 8,16:PRINT CHR$(7)"En
trez au maximum 8 caract[res":FOR i=1 TO 300:NEXT:PAPER #1,2
:GOTO 1500
1580 c$=",.:"+CHR$(34)
1590 FOR i=1 TO LEN(c$):IF INSTR (file$,MID$(c$,i,1))=0 THEN
 NEXT:GOTO 1610
1600 LOCATE 10,15 : PEN 3:PRINT CHR$(7)"N'utilisez pas (,.:"
CHR$(34)")":FOR i=1 TO 300:NEXT:PAPER #1,2:GOTO 1500
1610 RETURN
```

```
1620 :
1630 :
1640 REM ========================================
1650 REM    Entr\e du fichier
1660 REM ========================================
1670 :
1680 GOSUB 1490 : IF file$="*ESC*" THEN RETURN ELSE GOSUB 10
20 : IF found=0 THEN 1680
1690 ef=0 : OPENIN file$+".dat" : IF ef THEN 1690
1700 INPUT#9,nombrechamps
1710 ERASE d$:DIM d$(200,nombrechamps-1)
1720 FOR i=0 TO nombrechamps-1
1730  INPUT#9,m$(i)
1740 NEXT
1750 INPUT#9,nombre
1760 FOR i=1 TO nombre
1770   FOR i1=0 TO nombrechamps-1
1780      INPUT#9,d$(i,i1)
1790   NEXT i1
1800 NEXT
1810 CLOSEIN
1820 RETURN
1830 :
1840 REM ==============================
1850 REM    Fin du programme
1860 REM ==============================
1870 :
1880 |USER,0 : CALL desactive : MODE 2 : PRINT "*** Fin du p
rogramme ***" : END
1890 :
1900 REM ==============================
1910 REM    Affiche contenu disquette
1920 REM ==============================
1930 :
1940 MODE 2 : CAT
1950 LOCATE 36,25 : PRINT "<< Frapper une touche >>"
1960 CALL &BB06 : RETURN
```

```
1970 :
1980 REM ============================
1990 REM   Modifier/Cr\er masque
2000 REM ============================
2010 :
2020 MODE 1 : WINDOW #1,1,40,1,3 : PAPER #1,2 : INK 2,1 : CL
S#1 : LOCATE #1,7,2 : PRINT #1,"Cr\er et modifier masque
2030 LOCATE 1,7 : IF nombre>0 THEN jusque=nombrechamps-1 ELS
E jusque=9
2040 FOR i=0 TO jusque
2050  PRINT "Champ "RIGHT$(STR$(i+1),2)":"m$(i)
2060 NEXT
2070 WINDOW #1,5,35,23,25 : PAPER #1,2 : PEN #1,1 : CLS#1 :
LOCATE #1,8,2 : PRINT#1,"Ctrl/Enter pour fin
2080 WINDOW #1,10,40,7,7+jusque : PAPER #1,0 : PEN #1,1
2090 CLS#1:FOR I=0 TO nombrechamps-1:PRINT #1,M$(I)
2100 NEXT:LOCATE #1,1,nombrechamps+1+(nombrechamps=jusque+1)
:i2=VPOS(#1)-2:LINE INPUT #1,"",a$ : IF a$="*ESC*" THEN RETU
RN ELSE IF a$="" THEN LOCATE #1,1,nombrechamps+1 : GOTO 2090

2110 i1=VPOS(#1)-2:IF i2=8 AND i1=8 THEN i1=9
2120 m$(i1)=a$ : IF i1>nombrechamps-1 THEN nombrechamps=i1+1

2130 GOTO 2090
2140 :
2150 REM ====================================
2160 REM    Entr\e/modification des donn\es
2170 REM ====================================
2180 :
2190 IF nombrechamps=0 THEN RETURN ELSE MODE 2 : WINDOW #1,1
,80,1,3 : PAPER #1,1 : PEN #1,0 : CLS #1
2200 IF stat=0 THEN a$="Entr\e et modification de donn\es" E
LSE IF stat=3 THEN a$="S\lection de l'enregistrement { suppr
imer" ELSE IF stat=5 THEN a$="S\lection des enregistrements
{ sortir"
2210 LOCATE #1,40-LEN(a$)/2,2:PRINT #1,a$
2220 GOSUB 2580
2230 LOCATE 1,7 : FOR i=0 TO nombrechamps-1
2240   PRINT m$(i)TAB(longueur-2)":"
2250 NEXT
2260 LOCATE 60,5 : PRINT "Longueur du champ de donn\e="80-lo
ngueur
```

```
2270 PEN #2,0 : PEN #3,1 : PAPER #2,1 : PAPER #3,0
2280 WINDOW #2,5,75,22,23 : CLS #2 : IF stat<5 THEN PRINT #2
,TAB(7)"** Touches Ctrl/Enter=Changer \tat (entr\e/correctio
n) **":IF stat=0 THEN a$="** Entrez dans n'importe quel cham
p 'fin' pour fin **":GOTO 2300
2290 IF stat<>2 THEN a$="** Remplissez le masque avec le cri
t[re de recherche **" ELSE a$="** Entrez maintenant les corr
ections **"
2300 PRINT #2,TAB(35-LEN(a$)/2)a$
2310 WINDOW #1,1,80,25,25 : PAPER #1,1 : PEN #1,0
2320 CLS#1 : LOCATE #1,5,1 : PRINT #1,"Stock\ :";nombre : LO
CATE #1,55,1 : PRINT #1,"Etat : "; : IF stat<>1 THEN PRINT#1
,"** Entr\e **" ELSE PRINT#1,"** Correction **"
2330 PO=0
2340 IF stat<>2 THEN WINDOW #3,longueur,79,7,16 : CLS#3
2350 WHILE PO<nombrechamps : WINDOW#2,1,longueur-3,7+PO,7+po
 : WINDOW #3,1,longueur-3,7+po,7+po : CLS#2 : PRINT #2,m$(po
)
2360 WINDOW #4,longueur,79,7+po,7+po : LINE INPUT #4,a$ : a$
=LEFT$(a$,80-longueur)
2370 IF a$="*ESC*" THEN 2500 ELSE IF LOWER$(a$)="fin" THEN R
ETURN
2380 da$(po)=a$
2390 CLS#3:PRINT#3,m$(po):po=po+1
2400 WEND
2410 IF stat=0 THEN nombre=nombre+1+(nombre=200):FOR i=0 TO
nombrechamps-1:d$(nombre,i)=da$(i):NEXT:GOTO 2320
2420 IF stat=2 THEN 2470
2430 IF stat=5 THEN RETURN
2440 GOSUB 700 : REM d\termine cha`ne recherch\e
2450 LOCATE 25,20 : IF found=0 THEN PRINT CHR$(7)"** Cet enr
egistrement n'existe pas **" ELSE PRINT SPACE$(40)
2460 IF found=0 THEN 2320 ELSE FOR i=0 TO nombrechamps-1:LOC
ATE longueur,7+i:PRINT d$(found,i):NEXT:IF stat=3 THEN RETUR
N ELSE stat=2:GOTO 2280
2470 FOR i=0 TO nombrechamps-1:IF da$(i)="" THEN 2490
2480 d$(found,i)=da$(i)
2490 NEXT:stat=1:GOTO 2500
2500 IF stat=5 THEN RETURN ELSE IF st>1 THEN 2280 ELSE stat=
1-stat:GOTO 2280
2510 END
2520 RETURN
```

```
2530 :
2540 REM ================================
2550 REM  D\termine la plus grande cha`ne
2560 REM ================================
2570 :
2580 longueur=0
2590 FOR i=0 TO nombrechamps-1
2600   IF LEN(m$(i))>longueur THEN longueur=LEN(m$(i))
2610 NEXT
2620 longueur=longueur+4
2630 RETURN
2640 :
2650 REM ======================================
2660 REM      Supprimer un enregistrement
2670 REM ======================================
2680 :
2690 IF nombre=0 THEN RETURN ELSE stat=3 : st1=1 : GOSUB 219
0 : IF LOWER$(a$)="fin" THEN RETURN ELSE st1=0
2700 LOCATE 1,18 : PRINT "Enregistrement correct (o/n) ":EVE
RY 30,0 GOSUB 2740
2710 d$=INKEY$ : IF d$="" THEN 2710
2720 sn=REMAIN(0):IF LOWER$(d$)<>"o" THEN 2690
2730 sn=found : GOSUB 1110 : RETURN
2740 i=ABS(i=0):LOCATE 30,18:IF i THEN PRINT"?" ELSE PRINT"
"
2750 RETURN
```

```
2760 :
2770 REM ====================================
2780 REM     Trier
2790 REM ====================================
2800 :
2810 MODE 2:PRINT"Sur quel champ doit se faire le tri (1-"ST
R$(nombrechamps)"):"; : INPUT " ",a
2820 IF a<1 OR a>nombrechamps THEN 2810
2830 PRINT : PRINT "Champ "CHR$(34);m$(a-1);CHR$(34)" -- cor
rect ? (o/n)
2840 d$=INKEY$ : IF d$="" THEN 2840
2850 IF LOWER$(d$)<>"o" THEN 2810
2860 champ=a-1 : GOTO 840
2870 i9=(i9=0):LOCATE 5,7:IF i9 THEN PRINT" "CHR$(224) ELSE
PRINT CHR$(224)" "
2880 RETURN
2890 :
```

```
2900 REM ======================================
2910 REM     Sortie sur \cran et imprimante
2920 REM ======================================
2930 :
2940 MODE 2:PRINT"Sortie sur imprimante ou sur \cran ? (I/E)
"
2950 d$=INKEY$ : IF d$="" THEN 2950
2960 IF INSTR("ei",LOWER$(d$))=0 THEN 2950
2970 IF LOWER$(d$)="i" THEN ae=8 ELSE ae=0
2980 st1=1:stat=5:GOSUB 2190
2990 f=0
3000 FOR i=0 TO nombrechamps-1
3010   IF da$(i)<>"" THEN f(f)=i:f=f+1
3020 NEXT
3030 longueur=longueur-4:IF ae=0 THEN MODE 2
3040 FOR i=1 TO nombre
3050   FOR i1=0 TO f-1
3060     PRINT#ae,m$(f(i1))SPACE$(longueur-LEN(m$(f(i1))))":
 "d$(i,f(i1))
3070   NEXT
3080   PRINT#ae
3090 NEXT
3100 IF ae=0 THEN PRINT "<< Frapper une touche >>":CALL &BB0
6
3110 RETURN
```

```
3120 :
3130 REM ========================================
3140 REM    Routine d'erreur
3150 REM ========================================
3160 :
3170 IF ERR<>18 THEN RESUME NEXT
3180 ef=0 : CALL msg,@ds$
3190 nf=0:IF RIGHT$(ds$,9)="not found" THEN nf=1:RESUME NEXT
3200 WINDOW #6,1,40,24,25:PAPER #6,1:PEN #6,0:CLS #6
3210 PRINT #6,CHR$(7)"*** Disque:";ds$;" en";ERL:PRINT #6,"<
ENTER>=menu principal,sinon recommence
3220 d$=INKEY$ : IF d$="" THEN 3220
3230 ef=1 : IF ASC(d$)=13 THEN RESUME 510
3240 RESUME NEXT
```

5.5 DISKMON - UN MONITEUR DISQUETTE POUR LE CPC

Avez-vous essayé le programme de lecture et d'affichage de secteurs isolés de la disquette? Vous savez alors combien la sortie sur écran était lente avec ce programme. Dans le moniteur de disque suivant qui repose à l'origine sur le programme que nous venons d'indiquer, nous avons réalisé la sortie des informations de secteur dans un petit programme machine pour obtenir une vitesse de sortie acceptable. Nous avons également intégré d'autres instructions qui sont certainement très utiles dans certaines circonstances. Voici les différentes instructions dont vous disposez:

R-lire des secteurs
M-modifier le contenu d'un secteur
W-écrire des secteurs
C-afficher le catalogue
B-calculer les numéros de piste et de secteur à partir du numéro de bloc

La lecture des secteurs a été organisée de façon très pratique. N'importe quel secteur de la disquette peut être atteint avec les quatre touches curseur. Les touches 'curseur haut' et 'curseur bas' permettent de sélectionner respectivement la prochaine piste plus élevée ou moins élevée, les touches 'curseur gauche' et 'curseur droite' sélectionnent le secteur qui se trouve avant ou après le secteur actuel.
Un 'wrap around' est intégré à ces fonctions. C'est-à-dire que si vous avez lu le secteur 9 d'une piste et que vous appuyez ensuite la touche 'curseur droite', c'est le secteur 1 de la prochaine piste qui est affiché. Notez la particularité suivante: lorsque vous tenez enfoncée les touches curseur, ce n'est pas le contenu du secteur actuellement lu qui est affiché. La vitesse d'accès est ainsi nettement accélérée. Malheureusement il en résulte également que ce n'est pas toujours le dernier secteur lu qui est affiché, de sorte que nous vous conseillons de le lire à nouveau avec 'R(ENTER)'.
Vous pouvez naturellement également sélectionner directement un secteur précis. L'instruction 'R' sert aussi à cela. Mais au lieu d'ENTER entrez alors un numéro de lecteur, dans la forme 0 pour lecteur A ou 1 pour lecteur B, le programme vous demandera alors les valeurs de piste et secteur voulues. Le secteur sera alors lu et la première moitié du contenu de ce secteur sera affichée.

Il n'est malheureusement pas possible de faire entrer dans un écran la

totalité des 512 octets d'un secteur en dump ASCII et HEXA. C'est pour cette raison qu'on peut basculer entre les deux pages écran correspondant à un secteur avec les touches 'Shift curseur gauche' et 'Shift curseur droite'.

L'instruction 'M' permet de modifier les informations lues. Cette instruction vous demande l'adresse de buffer à modifier. Si vous répondez à cette question avec ENTER, c'est le début du buffer qui sera sélectionné comme adresse.
L'adresse est alors affichée en même temps que le contenu. Le programme attend ensuite la nouvelle valeur de l'adresse de buffer. Vous pouvez entrer les nombres hexa 00 à FF. Ces nombres doivent être entrés sans '&' et validés avec ENTER. La prochaine adresse de buffer est alors affichée avec le contenu correspondant. Si vous ne voulez toutefois pas modifier le contenu de l'adresse affichée, vous pouvez passer à l'adresse suivante en appuyant sur la touche ENTER. Vous pouvez sortir du mode d'entrée avec 'X(ENTER)'.

Vous pouvez ensuite réécrire sur la disquette un secteur ainsi modifié avec l'instruction 'W'. En entrant 'W(ENTER)', le secteur est écrit dans le secteur d'où il avait été lu à l'origine. Vous pouvez cependant choisir également un autre secteur en indiquant la piste et le secteur avec l'instruction 'M'.

L'instruction 'B' constitue en quelque sorte une aide à la lecture de fichiers. Comme vous l'avez vu dans les chapitres précédents, les numéros des blocs occupés par chaque fichier sont sauvegardés dans le catalogue. Il n'est pas difficile de convertir ces numéros de bloc en numéros de piste et de secteur, mais c'est tout de même assez pénible. Ce travail nous est épargné par l'instruction 'B'. Indiquez à cet effet le numéro de bloc trouvé dans le catalogue et le CPC calculera automatiquement pour vous les valeurs correspondantes et les affichera dans la ligne d'en-tête. Avec 'R(ENTER)' vous pouvez alors lire et faire afficher le premier secteur ainsi calculé du bloc. Vous obtenez le second secteur en appuyant sur la touche 'curseur droite'.

L'instruction 'C' affiche le catalogue normal de la disquette. Cette fonction est très utile car vous n'êtes pas obligé de quitter le programme pour examiner le contenu de la disquette.

Comme le programme est largement écrit en Basic, il peut aisément être étendu avec vos propres routines et extensions. Une extension possible

consisterait par exemple à afficher le format de la disquette placée dans le lecteur de disquette et à reconnaître un changement de disquette. Si vous avez en effet lu une disquette en format de données, vous devez, dans la version actuelle du programme, envoyer d'abord l'instruction 'C', en cas de changement de format, pour qu'AMSDOS reconnaisse le changement de format et modifie le DPB en fonction de cela. Vous pourriez résoudre ce problème en déconnectant les messages d'erreur et en examinant d'après les flags si le secteur a été lu avec succès. Si ce n'est pas le cas, vous pourriez alors sauter à la routine AMSDOS pour déterminer le format (&C56C) et lire ensuite le secteur une nouvelle fois.
Un autre point faible que vous pourriez éventuellement améliorer est constitué par l'instruction 'B'. Elle ne fonctionne correctement qu'avec les formats CP/M standard et Vendor. Les résultats calculés pour les autres formats sont faux.
Une autre amélioration possible consisterait à sortir le contenu du secteur sur imprimante. La routine hexdump de l'exemple de lecture d'un secteur isolé pourrait rendre de grands services à cet égard.

Mais malheureusement, avant que vous ne pensiez à modifier notre programme, il vous faut encore le taper. Mais cela ne devrait pas poser de problème, notamment grâce aux valeurs de contrôle pour les lignes de Datas.

```
1000 ' *****   MONITEUR DISQUETTE   *****
1010 ' *****   RBR 17/4/1985        *****
1020 '
1030 DEFINT A-L,N-Z
1040 MEMORY &A000-1
1050 MODE 2
1060 LOCATE 10,10:PRINT"Veuillez patienter ..."
1070 GOSUB 9000:'definir les fenetres et POKEr les routines
machine en memoire
1080 CLS#1
1090 ins$=" "
1100 buffer=&A2
1110 cmd$="CRWMB"+CHR$(&F0)+CHR$(&F1)+CHR$(&F2)+CHR$(&F3)
1120 cmd$=cmd$+CHR$(&F6)+CHR$(&F7)
1130 FOR i=1 TO LEN(cmd$):cmd1$=cmd1$+MID$(cmd$,i,1)+",":NEX
T
1140 cmd1$=LEFT$(cmd1$,LEN(cmd1$)-1)
1150 sector=1:track=0:drive=0:instruction=&84:catflag=0
1160 '
1170 '
2000 ' *****   PROGRAMME PRINCIPAL   *****
2010 '
2020 POKE &A108,buffer:POKE &A10A,buffer
2030 LOCATE #1,1,1:CLS
2040 GOSUB 6000:'afficher une page
2050 PRINT"INSTRUCTION "cmd1$" >"
2060 ins$=UPPER$(INKEY$):IF ins$="" THEN 2060 ELSE CLS
2070 ON INSTR(cmd$,ins$) GOTO 3100,3150,3200,3250,3300,3500,
3550,3600,3650,3400,3450
2080 GOTO 2050
2090 '
2100 '
3000 ' *****   TABLE D'INSTRUCTIONS   *****
3010 '
3100 '*** INSTRUCTION D
3110 GOSUB 4000:GOTO 2050
3120 '
3150 '*** INSTRUCTION R
3160 GOSUB 5000:GOTO 2020
3170 '
3200 '*** INSTRUCTION W
3210 GOSUB 5100:GOTO 2020
3220 '
```

```
3250 '*** INSTRUCTION M
3260 GOSUB 7000:GOTO 2030
3270 '
3300 '*** INSTRUCTION B
3310 GOSUB 8000:GOTO 2030
3320 '
3400 '*** AFFICHER LA PREMIERE PAGE
3410 POKE &A108,BUFFER:POKE &A10A,BUFFER:GOTO 2030
3420 '
3450 '*** AFFICHER LA SECONDE PAGE
3460 POKE &A108,BUFFER+1:POKE &A10A,BUFFER+1:GOTO 2030
3470 '
3500 '*** TRACK+1
3510 TRACK=-(TRACK<39)+TRACK+((TRACK=39)*39):GOSUB 5500:GOTO
 2020
3520 '
3550 '*** TRACK-1
3560 TRACK=(TRACK>0)+TRACK+(-(TRACK=0)*39):GOSUB 5500:GOTO 2
020
3570 '
3600 '*** SECTOR-1
3610 SECTOR=(SECTOR>1)+SECTOR+(-(SECTOR=1)*(PEEK(&A8A0+(DRIV
E*64))-1))
3620 IF SECTOR=9 THEN GOTO 3550 ELSE GOSUB 5500:GOTO 2020
3630 '
3650 '*** SECTOR+1
3660 SECTOR=-(SECTOR<9)+SECTOR+((SECTOR=9)*(PEEK(&A8A0+(DRIV
E*64))-1))
3670 IF SECTOR=1 THEN GOTO 3500 ELSE GOSUB 5500:GOTO 2020
3680 '
4000 '*** AFFICHER LE DIRECTORY
4010 WINDOW SWAP 0,1:CLS:PRINT:|DIR:WINDOW SWAP 1,0:CATFLAG=
1
4020 RETURN
```

```
4030 '
5000 '*** LIRE SECTEUR
5010 PRINT"        LIRE SECTEUR":PRINT
5020 GOSUB 5800:GOTO 5500
5030 '
5100 '*** ECRIRE SECTEUR
5110 INSTRUCTION=&85:PRINT"        ECRIRE SECTEUR":PRINT
5120 GOSUB 5800
5130 '
5500 '*** LIRE/ECRIRE SECTEUR
5510 POKE &A100,INSTRUCTION
5520 POKE &A104,DRIVE
5530 POKE &A105,TRACK
5540 POKE &A106,SECTOR-1+PEEK(&A89F+DRIVE*&40)
5550 CALL &A0A0:INSTRUCTION=&84:RETURN
5560 '
5800 '*** ALLER CHERCHER DRIVE, TRACK SECTEUR
5810 INPUT"        DRIVE (0/1) OU ENTER";DRIVE$:IF DRIVE$=""
THEN RETURN
5820 DRIVE=VAL(DRIVE$)
5830 INPUT"        TRACK (0-39)         ";TRACK
5840 INPUT"        SECTEUR (1-9)        ";SECTOR
5850 RETURN
5860 '
6000 '*** AFFICHER
6010 IF CATFLAG=1 THEN CATFLAG=0:CLS#1
6020 WINDOW SWAP 0,1
6030 PRINT USING "      |  DRIVE ###   |   TRACK ###   |   S
ECTOR ###  |";DRIVE,TRACK,SECTOR
6040 PRINT
6050 IF INKEY$<>"" GOTO 6070
6060 IF INSTR("BW",INS$)=0 THEN CALL &BB81:CALL &A000:CALL &
BB84
```

```
6070 WINDOW SWAP 1,0
6080 RETURN
6090 '
7000 '*** MODIFY
7010 PRINT"         MODIFIER BUFFER"
7020 INPUT"ADRESSE BUFFER ";BUADR$
7030 IF BUADR$="" THEN BUADR=0 ELSE BUADR=VAL("&"+BUADR$)
7040 MADR=BUADR+(BUFFER*256)
7050 PRINT HEX$(BUADR,4);"  ";HEX$(PEEK(MADR),2);"  ";
7060 INPUT NEWBYT$:NEWBYT$=UPPER(NEWBYT$)
7070 IF NEWBYT$="X" THEN RETURN
7080 IF NEWBYT$="" THEN GOTO 7110
7090 NEWBYT=VAL("&"+NEWBYT$):NEWBYT$=""
7100 POKE MADR,NEWBYT
7110 BUADR=BUADR+1:GOTO 7040
7120 '
8000 '*** CONVERTIR NUMERO BLOC
8010 PRINT"       CONVERTIR NUMERO BLOC EN TRACK ET SECTOR"
8020 INPUT"       NUMERO BLOC (EN HEXA)  ";BLOCK$
8030 BLOCK$="&"+BLOCK$:A=VAL(BLOCK$)
8040 SECTCNT=A*2+18
8050 TRACK=(A*2+18)\9
8060 SECTOR=(A*2+18) MOD 9 +1
8070 RETURN
```

```
8080 '
9000 '*** DEFINIR FENETRES
9010 WINDOW #0,1,80,20,25
9020 WINDOW #1,1,80,1,19
9030 '
9100 '*** DATAS POUR HEXDUMP/SECTOR I-O
9110 FOR I=&A000 TO &A0BC
9120 READ BYTE:POKE I,BYTE:S=S+BYTE:NEXT
9130 DATA &C3,&2D,&A0,&7C,&CD,&08,&A0,&7D,&F5,&1F,&1F,&1F,&1
F,&CD,&11,&A0
9140 DATA &F1,&E6,&0F,&C6,&30,&FE,&3A,&38,&02,&C6,&07,&C3,&5
A,&BB,&3E,&0D
9150 DATA &CD,&5A,&BB,&3E,&0A,&C3,&5A,&BB,&3E,&20,&C3,&5A,&B
B,&2A,&07,&A1
9160 DATA &ED,&5B,&09,&A1,&FD,&21,&01,&00,&06,&10,&E5,&C5,&E
D,&4B,&0B,&A1
9170 DATA &09,&CD,&03,&A0,&C1,&E1,&CD,&28,&A0,&CD,&28,&A0,&7
E,&CD,&08,&A0
9180 DATA &FD,&2B,&CD,&28,&A0,&A7,&ED,&52,&19,&28,&2C,&23,&1
0,&EE,&01,&F0
9190 DATA &FF,&09,&06,&10,&CD,&28,&A0,&7E,&F6,&7F,&FE,&20,&3
8,&02,&18,&02
9200 DATA &3E,&2E,&CD,&5A,&BB,&23,&10,&EF,&CD,&1E,&A0,&E5,&3
7,&ED,&52,&E1
```

```
9210 DATA &C8,&FD,&E5,&C1,&0D,&18,&B1,&05,&3E,&10,&90,&4F,&0
6,&00,&ED,&42
9220 DATA &23,&41,&18,&D0,&4E,&4B,&3A,&4C,&44,&09,&41,&2C,&2
2,&20,&22,&0D
9230 DATA &21,&00,&A1,&CD,&D4,&BC,&22,&01,&A1,&79,&32,&03,&A
1,&21,&04,&A1
9240 DATA &5E,&23,&56,&23,&4E,&21,&00,&A2,&DF,&01,&A1,&C9,&2
D
9250 IF S<>20027 THEN PRINT"ERROR IN CHECKSUM":END
9300 S=0
9310 FOR I=&A100 TO &A10C
9320 READ BYTE:POKE I,BYTE:S=S+BYTE:NEXT
9330 DATA &84,&00,&00,&00,&00,&00,&01,&00,&A2,&FF,&A2,&00,&5
E
9340 IF S<>806 THEN PRINT"ERROR IN CHECKSUM":END
9350 RETURN
```

5.6 LA GESTION DE DISQUETTE

La gestion de disquette sert à créer et à traiter des disquettes de travail. Les différentes fonctions vous sont proposées dans les points du menu de sorte que vous n'êtes plus par exemple obligé de charger CP/M pour formater une disquette. Après que vous ayez tapé le programme de gestion de disquette, sauvegardez-le d'abord sur disquette. Après que vous ayez lancé le programme, vous obtenez le menu suivant à l'écran:

1) Changer nom de fichier
2) Suppression de fichier
3) Formatage d'une disquette
4) Affichage contenu disquette
5) Copie de fichiers NON-PROGRAMMES
6) APPEND - Fusion de deux fichiers
7) Afficher contenu d'un fichier

9) Fin du programme

Pour sélectionner une point du menu, vous n'avez qu'à actionner la touche correspondante du clavier (sans ENTER). Vous arrivez instantanément dans la routine correspondante.
Pour donner plus de clarté à nos explications, nous allons traiter tous les points du menu les uns après les autres.

1) Changer nom de fichier

Ce point du menu vous permet de changer le nom d'un fichier. On vous demande le nom du fichier dont vous voulez changer le nom. Entrez le nom correspondant et actionnez la touche ENTER.
SI VOUS VOULEZ SORTIR D'UN POINT DU MENU, ACTIONNEZ LORS DE L'ENTREE D'UN CHAMP QUELCONQUE LA TOUCHE ENTER, SANS AUTRES INDICATIONS.
Après que vous ayez entré le nom d'un fichier, on vous demande le nouveau nom. Si vous voulez par exemple changer le nom d'un fichier de "A" en "B", entrez "A" comme ancien nom et "B" comme nouveau nom.
On vous demande de placer dans le lecteur de disquette la disquette sur laquelle se trouve le fichier dont vous voulez changer le nom. Après que vous ayez actionné une touche, le nom fichier correspondant sur la disquette est changé.

2) Suppression de fichier

Ce point du menu est le plus pratique du programme de gestion de disquette. Il sert à "nettoyer" les disquettes. Vous serez surpris par la quantité de programmes et de données qui peuvent s'accumuler assez vite sur une disquette. Cette routine vous permet d'éliminer facilement de vos disquettes les programmes et fichiers de données devenus périmés et inutiles. Après que le catalogue de la disquette ait été affiché à l'écran, vous pouvez "aller" sur les différents noms de fichiers. En actionnant la touche COPY, vous pouvez marquer un fichier comme fichier à supprimer, le champ correspondant clignote alors entre rouge et rose, ou vous pouvez supprimer à nouveau ce marquage. Lorsque vous avez marqué tous les fichiers que vous voulez supprimer, actionnez la touche ENTER. On vous demande alors encore si vous avez vraiment marqué tous les fichiers et si vous voulez vraiment supprimer ces fichiers. Après que vous ayez répondu positivement à ces questions, les fichiers marqués sur la disquette sont supprimés.

Cette routine s'est avérée très utile pour nous, pour "nettoyer" les disquettes, particulièrement parce qu'elle est moins sujette à erreur que l'instruction ERA, a$ avec laquelle on a vite fait de supprimer par mégarde un fichier.

3) Formatage d'une disquette

Ce point du menu est particulièrement intéressant car il vous évite d'avoir à charger CP/M pour formater une disquette. Cette routine de formatage est plus rapide que la routine FORMAT sous CP/M; elle ne contrôle toutefois pas si les pistes ont été "correctement" formatées. Nous n'avons cependant jamais eu de problème avec des disquettes que nous avions formatées avec cette routine.

Vous pouvez formater en format Vendor et en format Data only; le formatage IBM n'a pas été prévu.

Le format Vendor signifie que la disquette est formatée comme disquette système sur laquelle ne figure PAS CP/M et qu'on se contente de réserver les deux pistes supérieures. Il vous faudrait alors copier CP/M sur la disquette sous CP/M avec l'instruction SYSGEN.

Vous avez en règle générale intérêt à formater votre disquette en format Data only; vous pouvez ainsi utiliser la totalité de la capacité de la disquette pour y sauvegarder des programmes Basic et des données.

Après que vous ayez sélectionné le format, on vous demande de placer dans le lecteur la disquette à formater et d'actionner une touche.

4) Affichage contenu disquette

Le contenu de la disquette est affiché à l'écran. Vous pouvez utiliser par exemple ce point du menu pour vérifier si un fichier dont vous voulez changer le nom ou que vous voulez supprimer se trouve bien sur la disquette.

5) Copie de fichiers NON-PROGRAMMES

Vous ne pouvez pas copier des fichiers programmes avec cette routine mais elle vous permet par contre de copier des fichiers de données copiés avec cette routine. Vous pouvez choisir si le fichier source doit être copié sur la même disquette ou sur une autre. Si vous voulez copier le fichier sur la même disquette, vous n'êtes soumis à aucune contrainte de place mémoire, contrairement à ce qui se passe si vous voulez copier le fichier sur une autre disquette: comme dans ce dernier cas, en effet, le contenu du fichier doit être entrestocké dans la mémoire centrale, la place mémoire disponible constitue une certaine limite; vous pouvez copier au maximum 250 enregistrements. Si le fichier est trop long pour pouvoir ^etre copié, cela vous sera indiqué par un message à l'écran.

6) APPEND - Fusion de deux fichiers

Ce point du menu vous permet de faire un seul fichier à partir de deux fichiers. On vous demande les noms des deux fichiers que vous voulez relier entre eux: le premier fichier constituera également plus tard la première partie du nouveau fichier et le second fichier sera ajouté à sa suite. Le fichier objet est créé sur la même disquette sur laquelle se trouvent également les deux fichiers sources.

7) Afficher contenu d'un fichier

Ce point du menu a le même effet que l'instruction TYPE sous CP/M: le contenu d'un fichier ASCII est affiché sur l'écran. En appuyant sur une touche quelconque, vous pouvez arrêter l'affichage ou le faire redémarrer.
L'une ou l'autre de ces routines vous rendra certainement de bons services dans votre travail quotidien avec le lecteur de disquette.

```
10 '*****************************************
20 '*** Gestion de disquette ---- JS/RB 1.5.85 ***
30 '*****************************************
40 :
50 OPENOUT "dummy" : MEMORY HIMEM-1 : CLOSEOUT
60 DATA &3e,&00,&32,&2f,&80,&3a,&2f,&80,&57,&3a,&30,&80,&5f,
&3a,&31,&80
70 DATA &4f,&21,&35,&80,&df,&32,&80,&3a,&2f,&80,&fe,&27,&c8,
&3c,&32,&2f
80 DATA &80,&21,&35,&80,&06,&09,&77,&23,&23,&23,&23,&10,&f9,
&18,&d6,&27
90 DATA &00,&41,&52,&c6,&07
100 :
110 FOR i=&8000 TO &8034
120   READ d
130   POKE i,d
140   s=s+d
150 NEXT
160 IF s<>4258 THEN PRINT"*** Erreur en Datas ***"
170 MEMORY &7FFF
180 :
190 CLEAR : DEFINT b-z
200 MODE 1 : INK 0,11 : INK 1,16,6 : INK 2,0 : INK 3,24 : PE
N 3 : PAPER 2 : CLS
210 BORDER 0
220 CLS : ORIGIN 0,0,0,640,340,400 : CLG 3
230 PAPER 3 : PEN 2 : LOCATE 14,2 : PRINT"Gestion de disquet
te" : PAPER 2 : PEN 3
240 LOCATE 1,7
250 PRINT"1) Changer nom de fichier"
260 PRINT"2) Suppression de fichier"
270 PRINT"3) Formatage d'une disquette"
280 PRINT"4) Affichage contenu disquette"
290 PRINT"5) Copie de fichiers NON-PROGRAMMES"
300 PRINT"6) APPEND - Fusion de deux fichiers"
310 PRINT"7) Afficher contenu d'un fichier"
320 PRINT : PRINT"9) Fin du programme"
330 LOCATE 1,20 : PEN 1 : PRINT "Votre choix:"
340 a$=INKEY$ : IF a$="" THEN 340
350 we=VAL(a$) : IF we<1 OR (we>7 AND we<>9) THEN 340
360 PEN 3 : ON we GOTO 1200,550,380,1120,1340,1700,1900,1900
,2100
```

```
370 '
380 '================================
390 '     Formatage d'une disquette
400 '================================
410 :
420 CLS : PRINT"1) Format Vendor" : PRINT : PRINT"ou" : PRIN
T : PRINT"2) Format Data only"
430 a$=INKEY$ : IF a$<"1" OR a$>"2" THEN 430
440 IF a$="1" THEN f$="Vendor" : y=&41 ELSE f$="Data Only" :
 y=&C1
450 x=&8035
460 FOR i=1 TO 9
470   POKE x,0 : POKE x+1,0 : POKE x+2,y : POKE x+3,2
480   x=x+4
490   y=y+2 : IF (y AND &F) = &B THEN y=y-9
500 NEXT
510 PRINT : PRINT"Veuillez inserer la diquette" : PRINT"et f
rapper une touche..."
520 IF INKEY$="" THEN 520
530 CALL &8000 : GOTO 190
540 '
550 '===========================
560 '   Suppression de fichier
570 '===========================
580 :
590 DIM a$(65),era(64)  :CLS
600 LOCATE 4,11 : PRINT"Je lis le Directory ..."
610 PRINT : PRINT"    un moment, je vous prie
620 lin$=STRING$(40,154)
630 FOR i=0 TO 63
640   a$(i)=STRING$(11,32)
650 NEXT
660 a=PEEK(&BB5A) : POKE &BB5A,&C9 : CAT : POKE &BB5A,a
670 anz=PEEK(&A912) : a=PEEK(&A79C)*256 + PEEK(&A79B)+1
680 CLS
690 FOR i=0 TO anz
700   POKE @a$(i)+1,a-(INT(a/256)*256)
710   POKE @a$(i)+2,INT(a/256) : a=a+14
720 NEXT
730 FOR i=0 TO anz
740 IF ASC(LEFT$(a$(i),1)) = 0 THEN a=i : i=anz : GOTO 770
750 a$(i)=LEFT$(a$(i),8) + "." + RIGHT$(a$(i),3)
760 PRINT a$(i),
770 NEXT : anz=a
```

```
780 LOCATE 1,22 : PRINT lin$
790 txt$="Supprimer sur ce disque? (O/N)" : GOSUB 1090
800 GOSUB 1100 : IF LOWER$(a$)="o" THEN 840
810 txt$="ENTER = un autre disque, X = fin" : GOSUB 1090
820 GOSUB 1100 : IF a$=CHR$(13) THEN ERASE a$,era : GOTO 550
830 IF LOWER$(a$)="x" THEN 190 ELSE 820
840 txt$="Touche 'COPY' marque a supprimer" : GOSUB 1090
850 x=0 : xc=1 : yc=1 : GOTO 950
860 x=temp : GOSUB 1100 : IF a$=CHR$(13) THEN 1000
870 IF ASC(a$)=&F0 THEN x=x-3 : IF x<0 THEN 860 ELSE 950
880 IF ASC(a$)=&F1 THEN x=x+3 : IF x>anz-1 THEN 860 ELSE 950
890 IF ASC(a$)=&F2 THEN x=x-1 : IF x<0 THEN 860 ELSE 950
900 IF ASC(a$)=&F3 THEN x=x+1 : IF x>anz-1 THEN 860 ELSE 950
910 IF ASC(a$)<>&E0 THEN 860
920 era(x) = era(x) XOR 1 'Inverser le champ
930 LOCATE xc,yc : PAPER era(x)
940 PRINT a$(x); : PAPER 2 : GOTO 860
950 yco = x\3+1 : xco = (x- ((yco-1)*3))*13+1
960 LOCATE xc,yc : PAPER (era(temp)=0)*-2+era(temp)
970 PRINT a$(temp); : PAPER 2
980 LOCATE xco,yco : PAPER era(x) : PRINT a$(x); : PAPER 2
990 xc=xco : yc=yco : temp=x : GOTO 860
1000 LOCATE xc,yc : PAPER 2+(era(x)<>0) : PRINT a$(x); : PAP
ER 2 : txt$="Tous les fichiers sont marques? (O/N)" : GOSUB
1090
1010 GOSUB 1100 : IF LOWER$(a$)<>"o" THEN 840
1020 txt$="Vraiment supprimer (O/N)" : GOSUB 1090
1030 GOSUB 1100 : IF LOWER$(a$)<>"o" THEN 840
1040 txt$="Je supprime les fichiers !" : GOSUB 1090
1050 FOR i=0 TO anz
1060 IF era(i) THEN |ERA,@a$(i)
1070 NEXT
1080 GOTO 190
1090 LOCATE 4,24 : PRINT CHR$(20);txt$ : RETURN
1100 a$=INKEY$ : IF a$="" THEN 1100 ELSE RETURN
1110 '
1120 '================================
1130 '   Affiche contenu disquette
1140 '================================
1150 :
1160 CLS : CAT
1170 PRINT : PRINT"Frapper une touche ..."
1180 IF INKEY$="" THEN 1180 ELSE 190
1190 :
```

```
1200 '=================================
1210 '   Changement de nom de fichier
1220 '=================================
1230 :
1240 CLS
1250 INPUT "Ancien nom du fichier : ",falt$ : IF falt$="" TH
EN 190
1260 INPUT "Nouveau nom du fichier : ",Neu$ : IF neu$="" THE
N 190
1270 PRINT : PRINT"Veuillez inserer la disquette"
1280 PRINT : PRINT"sur laquelle se trouve le fichier"
1290 PRINT : PRINT"et actionnez une touche"
1300 IF INKEY$="" THEN 1300
1310 |REN,@neu$,@falt$
1320 GOTO 190
1330 '
1340 '=================================
1350 '   Copie de fichiers pas PRG
1360 '=================================
1370 :
1380 CLS : INPUT "Nom du fichier source : ",quell$ : IF quel
l$="" THEN 190
1390 INPUT "Nom du fichier objet  : ",ziel$ : IF ziel$="" TH
EN 190
1400 INPUT "Copier sur une autre disquette (O/N) : ",jn$
1410 IF LOWER$(jn$)="n" THEN 1620
1420 DEFSTR a : DIM a(250)
1430 PRINT : PRINT"Veuillez inserer la disquette source"
1440 PRINT"et actionnez une touche"
1450 IF INKEY$="" THEN 1450
```

```
1460 OPENIN quell$
1470 WHILE NOT EOF
1480   LINE INPUT #9,a(xc)
1490   xc=xc+1 : IF xc>250 OR FRE(0)<2000 THEN PRINT CHR$(7)
"Le fichier est trop grand" : FOR i=1 TO 1000:NEXT:CLOSEIN:R
UN 190
1500 WEND
1510 CLOSEIN
1520 PRINT : PRINT CHR$(7)"Veuillez inserer la disquette obj
et"
1530 PRINT"et actionnez une touche"
1540 IF INKEY$="" THEN 1540
1550 PRINT : PRINT"Je copie"xc"enregistrements."
1560 OPENOUT ziel$
1570 FOR i=0 TO xc
1580   PRINT #9,a(i); : IF LEN(a(i))<255 THEN PRINT #9
1590 NEXT
1600 CLOSEOUT
1610 RUN 190
1620 OPENIN quell$ : OPENOUT ziel$
1630 WHILE NOT EOF
1640   LINE INPUT #9,a$
1650   PRINT #9,a$
1660 WEND
1670 CLOSEIN : CLOSEOUT
1680 GOTO 190
1690 '
1700 '================================
1710 '    Fusion de deux fichiers
1720 '================================
1730 '
1740 CLS : INPUT "Nom du premier fichier : ",f$(0) : IF f$(0
)="" THEN 190
```

```
1750 INPUT "Nom du second fichier: ",f$(1) : IF f$(1)="" THE
N 190
1760 INPUT "Nom du fichier objet    : ",f3$ : IF f3$="" THEN
 190
1770 PRINT : PRINT"ok"
1780 OPENOUT f3$
1790 FOR i=0 TO 1
1800   OPENIN f$(i)
1810   WHILE NOT EOF
1820     LINE INPUT #9,a$
1830     PRINT #9,a$
1840   WEND
1850   CLOSEIN
1860 NEXT
1870 CLOSEOUT
1880 GOTO 190
1890 '
1900 '=============================
1910 '  Afficher contenu fichier
1920 '=============================
1930 :
```

```
1940 ON ERROR GOTO 2090
1950 CLS : INPUT "Nom du fichier : ",f$ : IF f$="" THEN 190
1960 MODE 2 : PEN 1 : INK 1,1 : PRINT"Contenu du fichier "f$
1970 PRINT STRING$(80,"-")
1980 OPENIN f$
1990 WHILE NOT EOF
2000   LINE INPUT #9,a$
2010   PRINT a$
2020   IF INKEY$="" THEN 2040
2030   IF INKEY$="" THEN 2030
2040 WEND
2050 CLOSEIN
2060 PRINT : PRINT"Frapper une touche"
2070 IF INKEY$="" THEN 2070
2080 GOTO 190
2090 PRINT"File lype Error - pas fichier ASCII  !!" : RESUME
 2060
2100 CLS : PEN 1 : PRINT"Fin du programme"
```

INDEX

Les numéros renvoient au chapitre concerné

D

E

F

K

L

M

N

O

P

R

S

T

U

V

W

DATAMAT AMSTRAD CPC 464 ET 664

GESTION DE FICHIERS PROFESSIONNELLE POUR TOUS EN FRANÇAIS (accents à l'écran et à l'imprimante, messages, documentation...).

Avec DATAMAT, vous pouvez créer, modifier, rechercher, calculer, annuler, trier et imprimer jusqu'à 4.000 fiches.

DATAMAT est utilisable immédiatement, même par un débutant. Des menus en FRANÇAIS garantissent une utilisation aisée.

DATAMAT permet de travailler en 40 ou 80 colonnes, avec un ou deux lecteurs de disques.

DATAMAT permet :
- la recherche avec ou sans index,
- la recherche multicritères,
- le tri des fiches en ordre croissant ou non, suivant critères,
- l'impression sur différents types d'imprimantes (voir manuel).

DATAMAT est écrit en langage machine, ce qui garantit une excellente rapidité d'exécution et d'utilisation.

DATAMAT est utilisable conjointement avec TEXTOMAT, le traitement de texte, pour réaliser vos MAILINGS...

DATAMAT est livré en version 464 et 664 sur la même disquette, avec une documentation en FRANÇAIS de 60 pages environ.

Système requis :
Version disquette :
- Un Amstrad CPC 664 ou
  un Amstrad 464 avec unité de disquette DDI.
- Une imprimante (facultatif).

Réf. : AM304
Prix : 450 FF TTC

lab 372.26 91

TEXTOMAT AMSTRAD CPC 464 ET 664

TRAITEMENT DE TEXTE PROFESSIONNEL POUR TOUS EN FRANÇAIS (accents à l'écran et à l'imprimante, messages, documentation...).

Créez, modifiez, imprimez et archivez au bureau ou à la maison : courrier, mailing, documents, manuels, thèses, articles, rapports... mais de plus réutilisez tous ces textes ultérieurement en les modifiant si nécessaire.

TEXTOMAT est utilisable IMMEDIATEMENT même par un débutant. Un menu en FRANÇAIS en bas d'écran garantie une utilisation aisée.

TEXTOMAT intègre toutes les fonctions du traitement de texte (tabulation, recherche, remplacement, insertion, manipulation de paragraphes...).

TEXTOMAT comprend également des fonctions de CALCUL, de MAILING...

TEXTOMAT s'adapte à tout type d'imprimante grâce à une fonction particulière.

TEXTOMAT fonctionne en mode 80 colonnes.

TEXTOMAT affiche et imprime évidemment tous les ACCENTS de la langue française.

TEXTOMAT est écrit en langage machine ce qui assure rapidité et qualité du programme, celui-ci utilisant toutes les caractéristiques propres aux CPC 464 et 664.

Enfin TEXTOMAT peut relire les données de la gestion de fichiers DATAMAT, pour effectuer des mailings et des lettres type personnalisées.

TEXTOMAT est livré en version 464 et 664 sur la même disquette.

TEXTOMAT est donc bien la solution TRAITEMENT DE TEXTE pour AMSTRAD CPC.

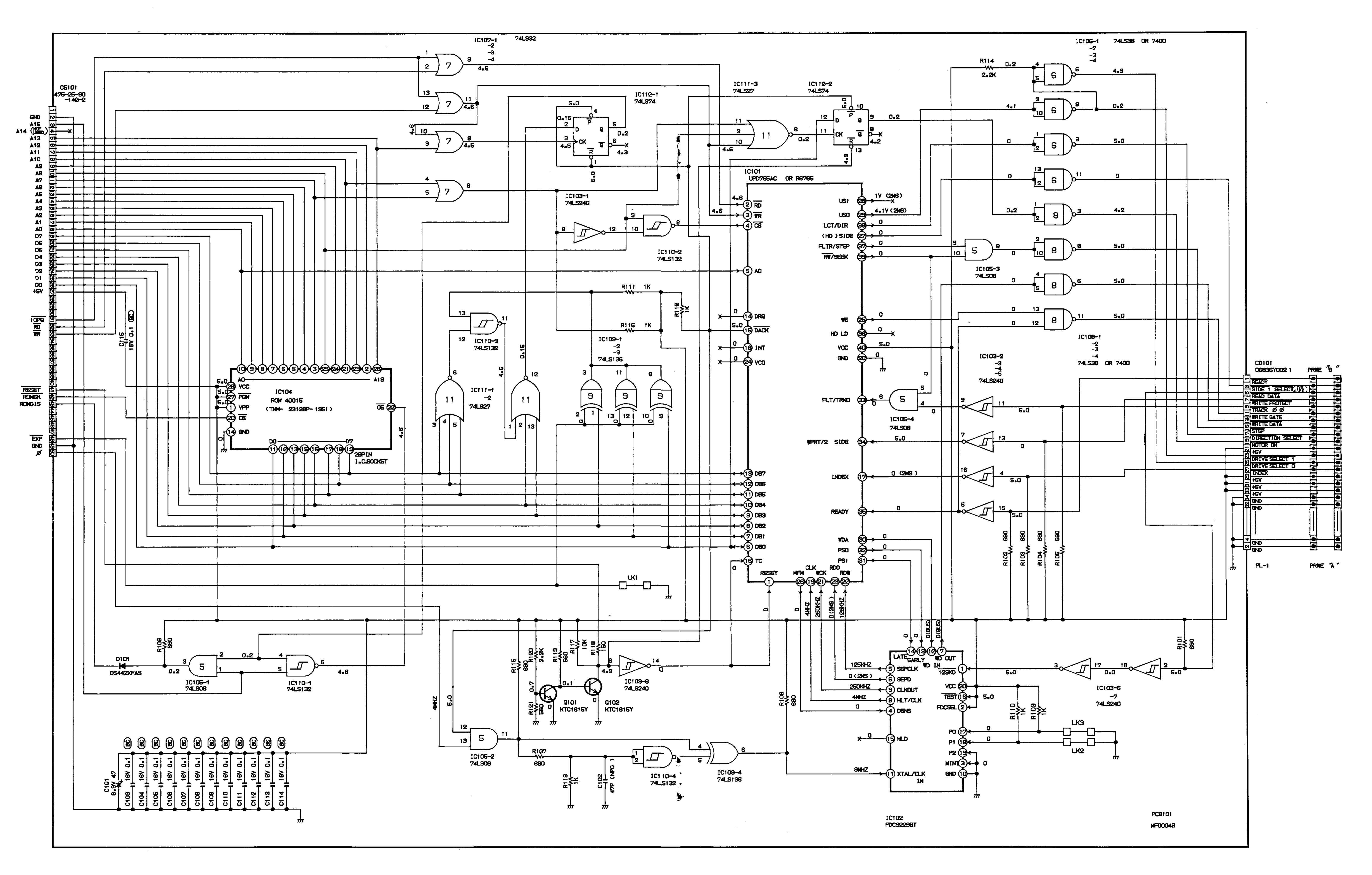
Système requis :
Version disquette :
- Un Amstrad CPC 664 ou
  un Amstrad 464 avec unité de disquette DDI.
- Une imprimante (facultatif).

Réf. AM305

Prix : 450 FF TTC

lab 372.26.91

Achevé d'imprimer en août 1985
sur les presses de l'imprimerie Laballery et Cie
58500 Clamecy
Dépôt légal : août 1985
Numéro d'imprimeur : 508010



NOTE: THIS SCHEMATIC DIAGRAM IS THE LATEST AT THE TIME OF PRINTING AND SUBJECT TO CHANGE WITHOUT NOTICE.

IMPRIMÉ EN FRANCE

[redacted]

TOUT SUR LE LECTEUR DE DISQUETTE AMSTRAD.

Tout sur la programmation et la gestion des accès disque avec l'AMSTRAD CPC 664 ou l'AMSTRAD CPC 464 et le FLOOPY DDI-1 ! Ce livre vous fournira de nombreuses informations et de précieux conseils ainsi que les listings d'utilitaires ultra performants comme un MONITEUR DISQUE, une GESTION DE FICHIERS RELATIFS... Vous trouverez également le LISTING du DOS commenté, la description électronique de l'appareil et une gestion de fichier complète. De nombreux exemples accompagnent chaque chapitre.

Aperçu des différents sujets traités :
— Tout sur les fichiers séquentiels
— Listing d'une gestion de fichier
— Explication des messages d'erreurs
— Gestion de fichiers relatifs (!)
— Listing du DOS commenté
— Programmation des contrôleurs
— Moniteur-Disque
— Gestionnaire de Disque
— Programmation en Assembleur
— Utilisation de CP/M

Ce livre est le must absolu pour tous les heureux possesseurs d'un lecteur de disquette AMSTRAD.

ISBN : 2-86899-022-3

Prix : 149 F TTC
Réf. : ML 127


AMSTRAD
LE LIVRE DU LECTEUR
DE DISQUETTE AMSTRAD
10



Document restauré et retravaillé à partir d'un scan de mauvaise qualité.
AMSTRAD
CPC
MÉMOIRE ÉCRITE
ACME
https://acpc.me